Mining Human Mobility Patterns from Pervasive Spatial and Temporal Data

A thesis in fulfilment of the requirements for the degree of Doctor of Philosophy

Amin Sadri
Master of Science (M.Sc.) - Communication Systems
(Iran University of Science and Technology)

School of Science
College of Science, Engineering and Health
RMIT University

March, 2018

Declaration

Amin Sadri
School of Science
College of Science, Engineering and Health
RMIT University
March, 2018

Acknowledgments

I would like to express my sincere gratitude and appreciation to a group of people that have supported my research in many ways.

First, I would like to thank my supervisor, Dr. Flora Salim. If it was not for her, I would not have come to Australia to study and would not have a Ph.D. degree. Flora has incredible vision and boundless energy supervising her Ph.D. students. When starting the Ph.D. in Australia, I had some issues with my visa application due to the sanctions imposed on my country, Iran. It took me 6 months to get the final approval and during that time, Flora patiently followed up my application until it was approved. She trusts my abilities and I felt this several times through the last four years when she gave me plenty of flexibility in my research. In summary, she was an awesome companion on my Ph.D. journey. Thank you!

I would also like to express my sincere gratitude to my associate supervisors, Dr. Yongli Ren and Prof. Timos Sellis for their advice and support. As an early researcher, I learned many things from Yongli as he spent quite a bit of time providing feedback on my work.

I would also like to thank my father (RIP), Mohammad and my mother, Farkhondeh. I always believe that I was lucky to be born into a family that encourages and supports me and my education. I know being born into such a family is much more important than my effort and my talents. I would also like to thank my sisters, Soraya and Samira. Despite being thousands of kilometres away, my success makes them all happy as always.

A special thanks to my beloved wife, Mahsa, who has been a constant companion on this journey. Thanks for the support, encouragement, patience, unwavering love and care that I continuously received from her while she herself was doing a Ph.D. as well. My little daughter, Saina, was born during my Ph.D. She patiently allowed me to do research. Sorry Saina if you spent extra time in childcare because both of us were doing Ph.D.s!

I would like to express my gratitude to all my friends in Iran and in Australia that directly or indirectly supported me throughout this time. Travelling to Iran and visiting family and lovable friends annually made me feel fresh for continuing my Ph.D. I would like to express my gratitude to my colleagues in the CRUISE group: Irvan, Saiedur, Jonathan, Hui, Rumi and Sam. They were part of my Ph.D. journey and made RMIT a better place for me through their friendship and research collaboration.

Finally, I would like to thank my scholarship provider, Sustainable Urban Precincts Program (SUPP), RMIT and Australia for giving me the opportunity to study for a Ph.D. and develop my career. I hope this piece of work will benefit the research community in my area.

List of Publications

- (Submitted) A. Sadri, Y. Ren, and F. D. Salim, C. Mascolo, J. Krumm. What will you do for the rest of your day? An approach to partial human daily trajectory prediction. Interactive, Mobile, Wearable and Ubiquitous Technologies, IMWUT 2018. (will be presented in CORE Rank A* conference, Ubicomp)
- A. Sadri, Y. Ren, and F. D. Salim. Summarizing movement graph for mobility pattern analysis. In Proceedings of the Knowledge Capture Conference, page 41. ACM, 2017. (CORE Rank A conference)
- A. Sadri, F. D. Salim, Y. Ren, M. Zameni, J. Chan, and T. Sellis. Shrink: Distance preserving graph compression. Information Systems, 69:180-193, 2017. (SJR Q1 journal- Impact Factor: 2.77)
- A. Sadri, Y. Ren, and F. D. Salim. Information gain-based metric for recognizing transitions in human activities. Pervasive and Mobile Computing, 38:92-109, 2017. (SJR Q1 journal- Impact Factor: 2.34)
- A. Sadri, Y. Ren, and F. D. Salim. Full trajectory prediction: What will you do the rest of the day? In Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct. ACM, 2017. (CORE Rank A* conference, Extended abstract paper)
- A. Sadri. Mining changes in mobility patterns from smartphone data. In Pervasive Computing and Communication Workshops (PerCom Workshops), pages 1-3. IEEE, 2016. (CORE Rank A* conference, workshop paper)
- A. Sadri and F. D. Salim. Day type classification using cell tower connectivity data from smartphones. In Proceedings of the 13th International Conference on Mobile and Ubiquitous Multimedia, pages 244-247. ACM, 2014. (CORE Rank B conference)

Contents

List of Figures

List of Tables

Abstract

Recent advances in communication, sensors and processors have made pervasive systems more computationally powerful and increasingly popular. These systems are deployed everywhere all the time while remaining transparent. Take smartphones as an example; they have become an integral part of human life and people carry them wherever they go. Coupled with the popularity of pervasive systems and user tracking, this has opened up excellent opportunities to analyse human mobility. This can be applied to a broad range of location-based services such as smart navigation and recommendation systems.

Data from pervasive systems has temporal, spatial and spatio-temporal aspects that can be leveraged for mining human mobility patterns. Temporal data such as time series from embedded sensors on smartphones does not usually have any information about locations, while time stamps are discarded in spatial data. The list of significant locations visited by the user is an example of spatial data. The third group of data is spatio-temporal data that has both temporal and spatial aspects such as users' trajectories. In this dissertation, we analyse human mobility by mining these three kinds of data. In each chapter, we look at a specific aspect to infer key information about users mobility including transition time detection, movement graph summarisation, and trajectory prediction.

We analyse temporal information from time series data to extract transition times in daily activities. The transition times denote when user activities change such as when the user goes to work or when the user goes shopping. In addition to applications in location-based services, extracting the transition times helps us to understand human mobility patterns across the whole day.

We tackle scalability to enable processing to take place on resource-constrained devices. We introduce Shrink as a new summarisation method to compress large scale graphs. Trajectories and movements of the user can be transformed into a graph in which each node represents stay points and each edge represents distance. Since this graph is very large,

Shrink is used to reduce the size of the movement graph while preserving distances between nodes. The property that is preserved in the compressed graph, also known as the coarse graph, is the distance between the nodes. Shrink is a query friendly compression, which means the compressed graph can be queried without decompression. As the complexity of distance-based queries such as shortest path queries is highly dependent on the size of the graph, Shrink improves performance in terms of time and storage. We also investigate the effect of compression on the human mobility mining algorithms and show that the summarisation provides a trade-off between efficiency and granularity.

We also analyse spatial-temporal data by predicting user trajectory based on historical data. Specifically, given the historical data and the users trajectory in the first part of the current day (e.g. trajectory in the morning), we predict how users will complete their trajectory in that particular day (e.g. predicting the trajectory for the rest of the day or the afternoon). The granularity of the predicted trajectory is the same as the granularity of the given trajectories. We emphasize that the predicted trajectory includes the sequence of future locations, the stay times, and the departure times. This enhances the user experience because by having the detailed trajectory in advance, location-based services can notify users about the consequence of the movement.

In summary, this thesis contains efficient algorithms that can be applied to diverse aspects of pervasive signals for mining human mobility. The new algorithms are aimed at problems in transition time detection, summarisation, and prediction. The solutions address the scalability issues and can work in big pervasive temporal and spatial data effectively and accurately.

Chapter 1

Introduction

Human tracking has become ubiquitous nowadays with explosive advances in sensor technology, communication and mobility technologies. This pervasive data is collected from different sources: smartphone sensors, wearable sensors, positioning systems, network traffic controllers, geo-referenced datasets, mobile computer logs, smartphone connectivity data, Bluetooth connectivity data, location-aware and wireless communication devices and many more. Mobile devices such as smartphones are very popular and rich in terms of data since they are embedded with various kinds of sensors. Geospatial technology is another source of data that provide us with pervasive signals. Modern geopositioning technologies are growing rapidly, which results in more geospatial information.

The pervasive systems enable us to capture and analyse human mobility for different purposes as they are deployed all the time and everywhere. Devices such as smartphones have become an indispensable gadget for everyone. Whether a businessman, student, or employee in any company, everyone carries at least one smartphone. As a result, the collection of movement data offers us a new source of information to study human mobility behaviours. Mobility data, nowadays, is collected on a very large scale and with a high level of precision without causing any disruption for people. Consequently, the information related to human movement behaviour has become widely available, contributing to research for mining mobility patterns.

In this thesis, we process various types of pervasive data ranging from sensor data to trajectory data to analyse human mobility and to discover hidden patterns. The data can be viewed from three different aspects:

1. **Temporal aspect**: From this aspect, we focus on temporal information such as when

dealing with time series from sensors.

2. **Spatial aspect**: The temporal information of data such as time stamps are discarded when focusing on this aspect. For example, when processing a map of the user's city, only the spatial aspect is taken into account.

3. **Spatio-temporal aspect**: By considering spatio-temporal aspects, temporal and spatial information are both considered. Processing the users trajectories is an example of mining spatio-temporal data.

1.1 Motivation

Understanding human mobility plays a key role in many context-aware applications such as targeted advertising and location-based services. Developing the science to analyse human activities and to describe how people move in their daily lives promises to address a wide range of challenges.

In social networks, mining human mobility is the foundation in many areas such as friend recommendation and community discovery [Cho et al., 2011] [Lancichinetti and Fortunato, 2009]. For friend recommendation, similarities in mobility patterns are investigated and the person with the most similar mobility pattern is recommended as a friend. For example, two students with the same routine have similar mobility patterns and could be recommended as friends. The similarity of mobility patterns could also be a way of discovering communities in social networks as people in a community usually share the same mobility pattern.

Another application of mining human mobility patterns is in location-based services [Lian et al., 2015] [Noulas et al., 2012]. Users interact with the services using mobile devices and tend to get recommendations based on their location [Lian et al., 2015]. Searching for specific destinations, finding good nearby places to eat, receiving suggestions depending on location and finding friends in our area are just a few applications of location-based services [Noulas et al., 2012]. Inferring the current location or predicting the future location improves the service and enhances the user's experience. For example, the user may want to receive a notification just after they leave home or ten minutes before going to a gym.

In urban computing, understanding the mobility pattern of a crowd or a citizen individually can assist city planners [Zheng et al., 2014]. Data containing human mobility information has been produced by sensing technologies and large-scale computing infrastructures in urban spaces, which has yielded a large range of knowledge and can help tackle challenges in a city when used correctly. For example, Rinzivillo et al. leverage human mobility data to address the problem of detecting *real* city boundaries, not boundaries defined by the government. The result can provide policymakers with optimal administrative borders for the city. In one study, the problem of optimal retail store placement in New York is addressed by collecting and mining human mobility data [Karamshuk et al., 2013]. Human mobility combined with other data sources such as urban geography can also help predict the ranking of residential real-estate in a city at a future time [Fu et al., 2014].

The underlying problems in a city's road network can be detected and addressed by mining city-wide human mobility data. For example, mining mobility pattern enhances traffic management by improving traffic forecasting [Ribeiro et al., 2014] [Kitamura et al., 2000]. Another application is improving public transportation systems. As urbanization keeps changing cities, the transport services have to adapt their routes continuously in order to continue to meet the mobility demands of citizens. Analysing the mobility of the citizens enables us to update the transport routes regularly to meet their demands[Bastani et al., 2011].

In the domain of health and safety behaviours, the spreading of biological viruses is affected by human mobility. By mining data on human mobility, it is possible to recognize the group of people that are at high risk and, as a result, reduce the number of infections [Colizza et al., 2007]. Disasters such as earthquakes cause large, abnormal human population movements for evacuation. Disaster management requires an understanding and predicting of these movements to plan effective humanitarian relief [Song et al., 2013].

Mining human mobility patterns also benefits location recommendation systems [Lian et al., 2015]. The problem is defined as follows: given a set of users and some information about their past location preferences, the goal is to identify and recommend a set of locations they are more likely to enjoy. Mining users' mobility patterns is an important stage and essential for having an effective recommendation system. Figure 1.1 shows an example where the importance of understanding the mobility pattern is highlighted. In this example, Restaurant A is closer to the user's location but an effective recommendation system enhances the user's experience by suggesting Restaurant B, which is on the user's way home.

1.2 Research questions

The main research objective of this thesis is to analyse human mobility through mining temporal and spatial data. To overcome the aforementioned research challenges, the following

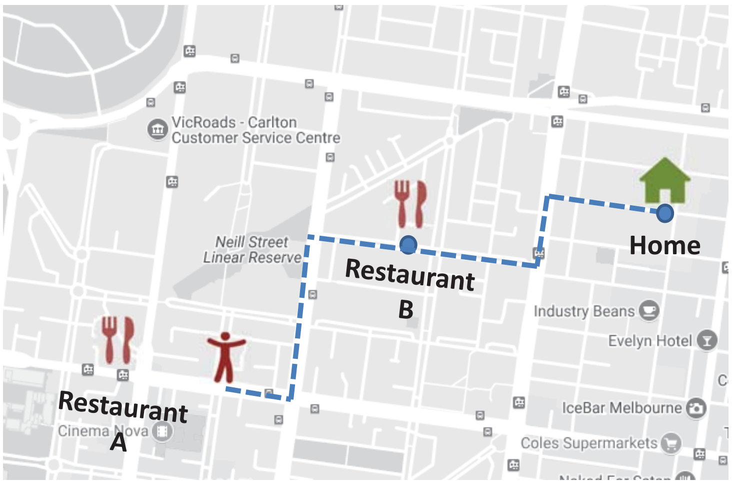


Figure 1.1: The recommendation system suggests Restaurant A while considering the users location only. However, having knowledge of the user's mobility pattern, Restaurant B is suggested because it is on the user's way home.

research questions are defined and addressed:

Temporal aspect

RQ1: Given multivariate time series from sensors, what is an appropriate temporal segmentation approach to extract transition times in human daily routine? For this research question, we analyse human mobility pattern by mining temporal data including time series from different types of sensors embedded in the smartphone.

Spatial aspect

RQ2: How to summarise graphs containing human movements in an efficient way? For this research question, the time stamps are discarded and we only focus on the locations and movements.

Spatio-temporal aspect

RQ3: Given the historical data and the user's trajectory in the first part of the current day (e.g. trajectory in the morning), how to predict the trajectory for the rest of the day? Here, both locations and time stamps are considered.

1.3 Challenges

Dealing with pervasive data and mining human mobility patterns both pose various challenges. First, the data is very heterogeneous and may be recorded in a variety of different formats ranging from time series to text logs and spatio-temporal data such as GPS data. Time series alone have many characteristics that vary from one case to another. The source of time series could be an accelerometer, gyroscope or any other sensor embedded in mobile devices. For example, temperature does not usually change rapidly, while the accelerometer signal changes very frequently from negative values to positive values and vice versa. Furthermore, the range, unit, and frequency of time series are different even for the same type of sensor depending on the manufacturer. One specific type of sensor probes temperatures between $0°$ to $30°$ while another temperature sensor is designed to report temperatures in the range of $0°$ to $50°$. Diversity in data makes mining difficult. As a result, we need to develop a generic method that can handle different types of data.

Another challenge is the diversity in human mobility patterns. Each person has a different mobility pattern compared to another person. For some people, it is hard to predict their daily activities as their activities change every day based on the user's decision. For example, a retired person or a housekeeper may decide what to do every morning and they do not have a strict routine. In contrast, other people's lives are characterized by strong patterns across all time scales such as an employee of a company that has strict working hours. Furthermore, some people visit a few places limited to one region in one day while others travel across the city for different purposes such as a taxi driver. Since a mobility pattern varies from one user to another, there should not be a presumption of the user's mobility before mining the data. The data mining models proposed in this area should be capable of handling all type of mobility patterns.

Every method proposed for mining human mobility should be scalable. Recent advances in communication, sensors and processors have provided a rich and large amount of the pervasive data for each individual including sensor data, connectivity data and smartphone logs. Furthermore, we sometimes apply a mining algorithm to a large group of people such as citizens of a city. In this case, the algorithm should be fast enough to respond in an acceptable time.

The methods applied to pervasive signals ideally need to be unsupervised rather than supervised. Supervised methods require data to be labelled. The labelling process not only requires time and energy but also is obstructively opposed to pervasive systems.

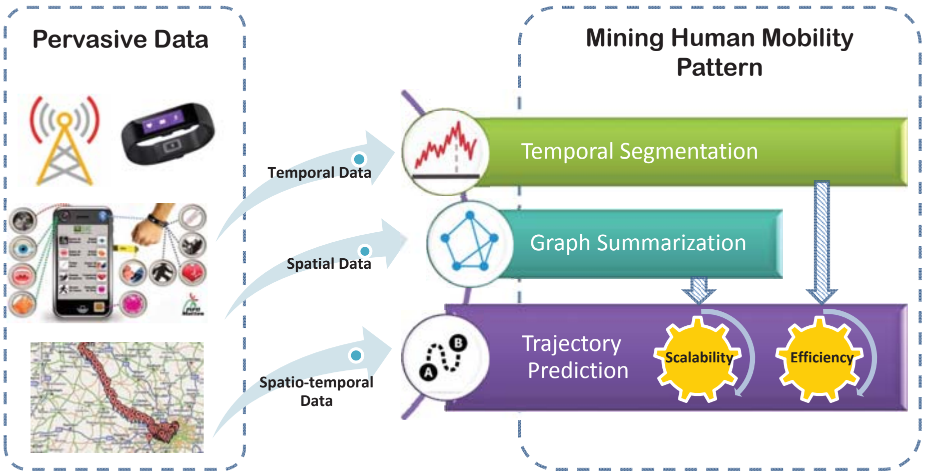


Figure 1.2: Relationships between the contributions of the thesis

1.4 Gaps and contributions

In this research, we propose new approaches for different problems in human mobility including temporal segmentation to detect transition times, graph summarisation to compress the mobility graph and trajectory prediction to complete the daily user trajectory. Each approach tackles a specific type of data including temporal data, spatial data and spatial-temporal data. Figure 1.2 shows an overview of the contributions.

1.4.1 Temporal segmentation for extracting the transition times

In Chapter 3, we analyse mobility patterns by extracting transition times in daily human activities. Extracting the transition times provides us with valuable information and helps us understand human mobility patterns across the whole day. For example, assume one leaves home at 7 am, works at the office from 9 am to 5 pm and goes outdoors until 12 am every day. With this interpretation, his daily routine can be expressed as home (7 am) → commuting (9 am) → office (5 pm) → outdoors (12 am) → home. [12 am–7 am], [7 am–9 am], [9 am–5 pm] and [5 pm–12 am] are the temporal segments in this example that convey semantic meaning from the routine.

To extract the transition times, we proposed a new temporal segmentation method that has practical features in comparison to the current methods. Most existing temporal segmentation methods are only applied to a single time series while our method is able to process multivariate time series. Moreover, the proposed temporal segmentation method is able to deal with heterogeneous data from different types of sensors. Unlike the existing information gain-based methods that cannot handle positively correlated time series, our method segments multivariate time series with or without correlation. Other state-of-the-art methods usually focus on a specific form of data and use specific characteristics of that data while the proposed method is generic and can process time series regardless of the type and frequency. As the number of the transition times in human activity is unknown, we also propose a heuristic method to detect the best candidate for the number of the segments. The complexity and computational cost of our algorithm are also investigated in Chapter 3.

1.4.2 Summarisation of movement graphs

Considering spatial aspects, we introduce Shrink in Chapter 4 as a new summarisation method to compress the graph that represents mobility data. Trajectories and movements of the user can be transformed into a graph in which each node represents stay points and each edge represent a distance. Since this graph is very large, Shrink is used to reduce the size of the movement graph while preserving distances between nodes. The compression is based on the iterative merging of the nodes in a way that the new weights have the least effect on the distances. The merging of the nodes continues until the desired size of the compressed graph is reached. As the complexity of distance-based queries such as shortest path queries is highly dependent on the size of the graph, Shrink improves performance in terms of time and storage. The approach has been applied to both weighted and unweighted graphs including road network, friendship network, collaboration network, web graph and social network. In the experiment, a road network with more than 2.5 million nodes is reduced by a fifth while the average relative error is less than 1%. We also investigate the effect of the graph summarisation on problems surrounding human mobility analysis.

Our graph summarisation method, Shrink, differs from the current methods in the following ways: (1) Shrink is developed for reachability and distance-based queries; (2) most methods are designed for unweighted graphs while Shrink can be applied to both unweighted and weighted graphs; (3) Shrink provides a compressed data structure, the coarse graph, that can be directly queried without decompression; (4) some compression methods only reduce the number of edges while Shrink reduces the number of nodes and edges; (5) Shrink can be performed incrementally for temporal graphs while most of the above methods are batch algorithms, requiring the decompression of the whole graph to perform minor changes such as edge insertion or node removal; and (6) Shrink not only specifies the distance between the two nodes but also provides the actual path containing laying nodes.

1.4.3 Partial trajectory prediction

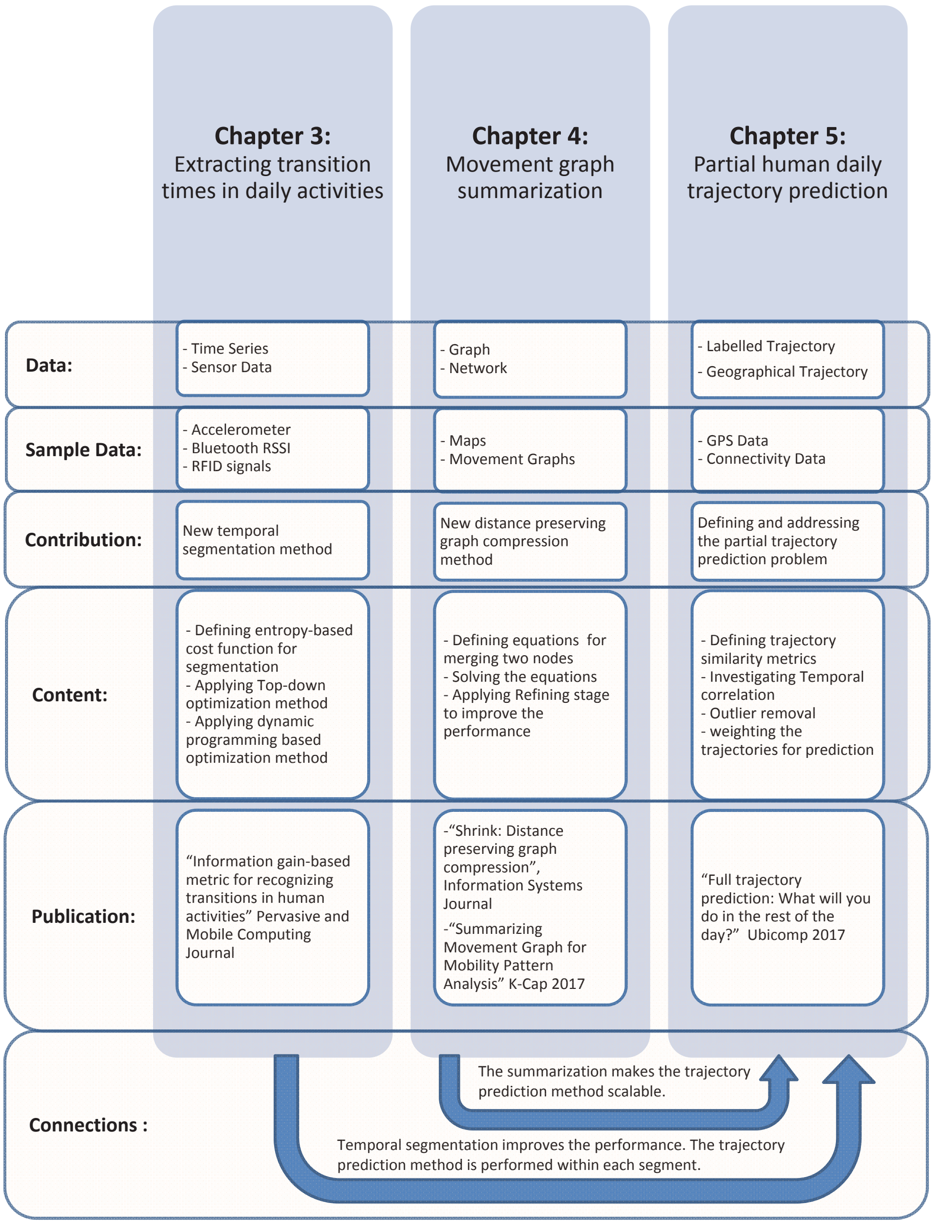
For analyses of the spatial-temporal aspects of the smartphone data, we predict the trajectory of the user based on the user's historical data. This problem is addressed in Chapter 5. Given the historical data and user's trajectory in the first part of the current day (e.g. trajectory in the morning), we predict the user's full movement by completing the trajectory for the rest of the day (e.g. prediction of the afternoon trajectory). We emphasize that the predicted trajectory includes the sequence of future locations, the stay times and the departure times. Furthermore, the granularity of the predicted trajectory is the same as the given daily trajectory.

Most of the existing research on human mobility prediction focuses on the prediction of the next location; a relaxed version of the trajectory prediction problem, which is the prediction of the place visited by the user at a certain time. Others predict a set of locations visited over a period of time without considering arrival/departure times or the sequence of the locations. Furthermore, existing methods pick a location only from significant locations (e.g. home and office) or stay points and the other locations are discarded, even for evaluation. Another important factor is the granularity of the predicted locations. The location prediction methods usually discretise the spatial data and return a sequence of regions. The discretization stage uses density-based clustering techniques to detect significant locations, stay points or points of interest (POIs). Some studies simply transform the trajectories into cells in the gridded map. While on the one hand discretization reduces the complexity of the problem and increases the certainty of the results, it also reduces the precision of the approach due to the coarse granularity of the regions. Another problem in human mobility is the prediction of departure times in which the lengths of the stays or dwell times are estimated.

Despite the importance of the knowledge contained in the user's trajectory, none of the existing research on human mobility focuses on the prediction of the trajectory that completes the users currently available daily trajectory. This trajectory includes the sequence of future locations with time stamps that will be visited by the user for the rest of the day. Specifically, it consists of spatio-temporal points such as GPS (Global Positioning System) coordinates with time stamps. The information embedded in the predicted trajectory includes 1) the geographic properties of the locations (e.g. latitude and longitude), 2) the sequence of the predicted locations (i.e. which place is visited first) and 3) the duration of the stays and departure times.

1.5 Thesis Organization

This chapter mainly discusses the challenges and motivation behind mining human mobility. The rest of this dissertation is structured as follows. In the next chapter, we provide the necessary background on different aspects of pervasive signals and human mobility mining. The main contributions of this dissertation are included in Chapters 3 to 5 as shown in Figure 1.3. In Chapter 3, a novel temporal segmentation method for extracting the transition times is presented. In Chapter 4, we address the movement graph summarisation by introducing a new distance preserving graph compression. In Chapter 5, we predict the user's trajectory containing the sequence of locations and transition times. Finally, Chapter 6 concludes the thesis with a summary of the major findings of the research conducted in each of the chapters and suggests developments of new approaches for future work.



	Chapter 3: Extracting transition times in daily activities	**Chapter 4:** Movement graph summarization	**Chapter 5:** Partial human daily trajectory prediction
Data:	- Time Series - Sensor Data	- Graph - Network	- Labelled Trajectory - Geographical Trajectory
Sample Data:	- Accelerometer - Bluetooth RSSI - RFID signals	- Maps - Movement Graphs	- GPS Data - Connectivity Data
Contribution:	New temporal segmentation method	New distance preserving graph compression method	Defining and addressing the partial trajectory prediction problem
Content:	- Defining entropy-based cost function for segmentation - Applying Top-down optimization method - Applying dynamic programming based optimization method	- Defining equations for merging two nodes - Solving the equations - Applying Refining stage to improve the performance	- Defining trajectory similarity metrics - Investigating Temporal correlation - Outlier removal - weighting the trajectories for prediction
Publication:	“Information gain-based metric for recognizing transitions in human activities” Pervasive and Mobile Computing Journal	-“Shrink: Distance preserving graph compression”, Information Systems Journal -“Summarizing Movement Graph for Mobility Pattern Analysis” K-Cap 2017	“Full trajectory prediction: What will you do in the rest of the day?” Ubicomp 2017

Connections :

The summarization makes the trajectory prediction method scalable.

Temporal segmentation improves the performance. The trajectory prediction method is performed within each segment.

Figure 1.3: Overview of the thesis structure

Chapter 2

Background

The analysis and mining of human mobility have a history spanning more than two centuries. The first work in this area belongs to Ravenstein et al., in which a set of principal laws that govern human migration is studied [Ravenstein, 1885]. From then until the advent of smart devices, the primary source for the study of human movement was surveys. As discussed in Chapter 1, recent advances in communications, sensors and processors enable the recording of human mobility through smart devices that generate various pervasive signals.

In this chapter, we provide some background on human mobility mining while focusing on each aspect separately. For temporal aspects, we discuss typical time series problems and applications. From the spatial point of view, we consider movement graphs as a way of representing human mobility and visits. Finally, for spatio-temporal aspects, we present the background on trajectory mining, including definitions and motivations. In each of the following three sections, we focus on a single aspect of pervasive data to address a problem in human mobility mining.

2.1 Time series mining

A time series is a sequence of data points that are typically ordered by time and each data point represents a value. In other words, time series data is a collection of observations with the timestamps measured at successive time intervals. Time series are available in many applications ranging from health to financial applications. Electrocardiograms, electroencephalograms and gene expression data are a few examples of the time series generated and used in the health domain alone. In addition, time series are widely used in industry, entertainment and finance [Keogh and Kasetty, 2003].

Table 2.1: Time series: algorithms, descriptions, and applications

Algorithms	Description	Sample applications
Query by Content	Given a query time series, and some similarity/dissimilarity measure, find the most similar time series in database DB	-Tool for exploratory data analysis -Important element of other time series algorithms
Clustering	Find natural groupings of the time series in a database under some similarity/dissimilarity measure	-DNA analysis (Bioinformatics) -Discovering similar trends in finance
Classification	Given an unlabelled time series, assign it to one of two or more predefined classes	-Braincomputer interface based on EEG signals -Human activity recognition from smartphone data
Segmentation (Change point detection)	Given a time series, partition it into several internally homogenous sections.	-Change in electricity consumption -Change human posture detection
Anomaly Detection	Given a time series find all sections which contain surprising, interesting, or unexpected occurrences	-Network intrusion detection systems -Anomaly consumption detection
Prediction (Forecasting)	Given a time series, predict the upcoming values of the time series in the future.	-Tourism and citizens demand forecasting -Medical surveillance

The ubiquity and wide use of time series have motivated researchers to mine and explore time series data. As a result, many new algorithms have been proposed for representation and indexing [Keogh et al., 2001a], similarity measure [Chen et al., 2007], segmentation [Keogh et al., 2004], classification [Xi et al., 2006], clustering [Liao, 2005] and visualisation of the time series [Kumar et al., 2005]. Most of these methods usually use a high-level representation of the data rather than the raw data. Table 2.1 shows the different applications of the time series in different areas [Fu, 2011].

Smartphones, smartwatches and wearable devices also generate time series in different ways [Lara and Labrador, 2013]. The embedded sensors are a common source of time series [Su et al., 2014]. Here, we briefly mention some of the common sensors. The smart devices come equipped with magnetometers that act as a compass and are able to sense magnetic fields. Gyroscope sensors measure angular velocity around three axes and determine if the device is twisted in any direction. Accelerometers also provide three-dimensional time series, reporting device acceleration in any given linear direction. The closeness of the device to an outside object is measured by proximity sensors. Barometers, ambient light sensors and fingerprint sensors are other types of sensors that are usually available in a gadget. In addition to the sensors, information about the status of the smart device such as connectivity data or logs can be collected in the form of time series. For example, considering smartphone connectivity data, the signal strength can be reported continuously in a time series format. The number of calls in a day, number of running applications and battery level are other examples of capturing the status of the smartphone in the form of time series [LiKamWa et al., 2013].

Although time series do not include spatial information inherently, user mobility can be inferred from the time series [Sun et al., 2014] [Farrahi and Gatica-Perez, 2008]. Sensors such as accelerometers and gyroscopes can be used to detect user activity related to their mobility such as walking or running. If this information is combined with the magnetometer, the direction of the movement can be inferred. Time series from other sources are also informative if the relationship between user mobility and those time series are investigated. For example, we may infer that the user does not call anyone during working hours in the office. Therefore, history of calls is an appropriate source to detect user location.

2.2 Representation of spatial aspect of mobility

The spatial aspect of a user's mobility includes locations visited by the user. This information can be represented in different ways. One common way of representing the spatial aspect of the user's mobility is to build a movement graph [Gambs et al., 2012] [Zheng et al., 2010]. In the movement graph, each node represents a location such as a point of interest (POI), a region covered by a cell tower ID (CID), a region covered by a WiFi access point, suburb, region defined by geographic coordinates, stay point[1] or road intersection. Similarly, each edge between two nodes can denote different parameters between the nodes such as distances, traveling times or frequencies of commutes [Giannotti et al., 2011].

The way the movement graph is defined depends on the data and problem. For example, Zheng et al. create a graph to extract travel sequences in which each node represents a cluster of stay points and each edge represents the frequency of the commutes between the nodes [Zheng et al., 2009]. In another piece of research, Hsu et al. use the weighted waypoint mobility model, in which the directed edges denote probabilities of choosing a destination based on the users current area. Therefore, the graph not only presents the locations that the user visits but also gives information about the relationship between the locations [Hsu et al., 2005]. Figure 2.1 shows an example of a movement graph where each node represents a stay point and the edges denote the movements between the nodes.

[1]A stay point can be defined as the location where the user spends a significant amount of time (e.g. one hour)

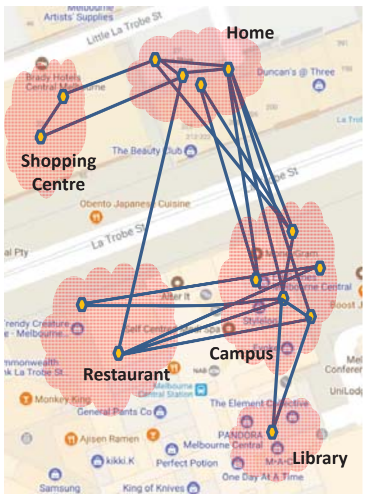


Figure 2.1: Sample movement graph of a user commuting between a shopping centre, restaurant, home, library and university

The size of the movement graph can be large or small depending on several factors. The first factor is the granularity of the locations. For example, if we only consider the POIs as the nodes of the graph, the user path consists of two nodes when the user goes from home to office. However, when the graph is the road network and the nodes represent the intersections, the user path would include more nodes. The second factor is the period of the time from which we collect data from the user. The user visits more places in a year compared to in a week. The last factor is related the user's behaviour. Some people visit more locations than others due to their lifestyle or their jobs. For example, the movement graph of a taxi driver would be much bigger than that of an employee in a company with set working hours. The processing and storage of the movement graph are challenging if the graph is large [Leskovec and Faloutsos, 2006] [Gubichev et al., 2010]. Furthermore, in some cases, processing becomes costly because the goal is to process not just one user but a group of users (such as people in a city) to analyse the behaviour of the group.

2.3 User trajectory mining

The mobility of the user can be directly presented by the trajectories that contain both spatial and temporal aspects. A *trajectory* is a trace in geographical spaces, represented by a series of chronologically ordered points, $p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_m$, where each point consists of a location and a timestamp $p = (loc, t)$. Thus, $< p_1, \cdots, p_i, \cdots, p_m >=<$

$(loc_1, t_1), \cdots, (loc_i, t_i), \cdots, (loc_m, t_m) >$. The trajectory can be categorised into two groups based on the information embedded in *loc*. The first group is geographical trajectories such as for GPS data, where *loc* is a geospatial coordinate set identifying the real location of the user. A GPS trajectory is an example of this group. The second group are the labelled trajectories; e.g. a sequence of Cell IDs of WiFi access points, and *loc* is the label assigned to a location. The labelled trajectories can be converted into geographical trajectories providing the mapping between the labels and the geographical coordinates. In some cases, mapping is not available due to privacy issues and therefore the trajectory is represented by a sequence of the labels where the location is unknown [Zheng, 2015].

The advances in location acquisition and mobile computing techniques have made trajectory data that can be captured in different ways increasingly popular. In an open area, a GPS receiver generates geographical trajectories containing the latitude and longitude of the location as well as timestamps. Indoor localisation techniques identify the user's location in a building. These techniques usually use WiFi signal to localise the user. Some methods use Bluetooth or RFID signals for localisation. In this case, the proximity to Bluetooth iBeacons or RFID tags in conjunction with timestamps define the trajectory.

From the connectivity data, the sequence of the WiFi access points connected to the smartphone can be considered as the labelled trajectory of the user. If the locations of the access points are known, the trajectory can be converted into a geographical trajectory. In addition to WiFi networks, the connectivity to the phone network can be used by reporting the connection to the CIDs. The granularity of the trajectory locations depends on the area that the access point or CID covers. Another way of capturing the users trajectory is by monitoring the activities that have logs on the smartphone. For example, the user transaction records reported on the smartphone when sorted chronologically also indicates the trajectory because each transaction contains the location where the transaction occurred and a timestamp. Another example is the check-ins of the user in a location-based social network that are accessible on smartphones [Song et al., 2010].

Mining the trajectories, which record the movement of a user, we can detect hidden mobility patterns and predict the locations that will be visited by the user. A lot of research confirms that human trajectories show a high degree of temporal and spatial regularity [Gonzalez et al., 2008] [Song et al., 2010]. Consequently, it is possible to generate theories and models of human mobility behaviour and use them for prediction [Giannotti et al., 2007].

2.4 Routine and predictability in human mobility

Recent research conducted by Song et al. has shown that humans are typically highly predictable in their movements, which is a key finding in this area [Song et al., 2010]. Analysing the location traces of 50,000 mobile phone users, they show that location prediction accuracy cannot be higher than 93% on average given the location of the users in the previous hour. A great deal of research attempts to reach this level of predictability [Scellato et al., 2011; Sadilek and Krumm, 2012; Bayir et al., 2010].

Many researchers have deployed the concept of entropy to measure the predictability in human behaviour [Pham et al.; McInerney et al., 2013]. In a study called reality mining, Eagle et al. use the entropy of cell tower connectivity distribution (BTS antennas) to measure the amount of predictable structure in an individual's life [Eagle and Pentland, 2006]. The predictability of the user's life can be quantified using entropy, which is typically measured in bits. People who have entropic lives tend to be more variable and harder to predict. In contrast, low-entropy lives are characterised by strong patterns across all time scales.

Eagle and Pentland extract the structure inherent in daily behaviour by using the principal component called eigenbehaviours. In this model, an individual's behaviour over a day is approximated by the sum of the eigenbehaviours gained from their historical behaviour. In fact, the eigenbehaviours are the eigenvectors of the covariance matrix of behaviour data, and the high-weighted vectors represent a type of repeated behaviours. The approach is applied to the reality mining dataset containing 100 subjects at MIT over the course of 9 months [Eagle and Pentland, 2009].

Sometimes, routine extraction is used for location and trajectory prediction. For example, Morzy et al. combine two well-known algorithms, PrefixSpan [Han et al., 2001] and FP-tree [Han et al., 2000], to discover moving rules of objects for prediction [Morzy, 2007]. Some methods extract frequent patterns from the historical data of all the users in the database and provide a global strategy that works for the prediction. The main assumption in these methods is that people tend to follow a crowd in their movements [Monreale et al., 2009; Chen et al., 2010; Lei et al., 2011]. For prediction purposes, using the frequent patterns approach alone does not always work because the user may not follow one of the frequent patterns.

2.5 Temporal aspects of human mobility mining

In some cases, only the temporal aspects of human mobility are considered and explored [Sun et al., 2014] [McInerney et al., 2013]. For example, some researchers focus on the prediction of departure times in which the lengths of stays or dwell times are estimated [Manweiler et al., 2013; Chen et al., 2012; Lee and Hou, 2006; Meng et al., 2015; Thajchayapong and Peha, 2006]. Specifically, the next location that the user goes to is not important but the time when the user leaves the current location is estimated.

McInerney et al. develop a mobility model that captures the tendency of users to depart from routine [McInerney et al., 2013]. First, they deploy an entropy estimator based on Lempel-Ziv to measure instantaneous predictability [Bhattacharya and Das, 1999]. Then, a Bayesian model is deployed to predict the time at which the user departs from a routine. They show that departures from routines correlate with mobile application usage; especially with the applications that provide information about local surroundings (e.g. maps).

2.6 Spatial aspects of human mobility mining

In some research questions about human mobility mining, it is mainly spatial aspects of mobility that are explored. Ashbrook et al. present a system that automatically clusters GPS data into significant locations [Ashbrook and Starner, 2003]. In some next location prediction methods, the time elapsed in the current location is not taken into account and it is only the next location that the user goes to that is predicted, be it after a short time or a long time [Gambs et al., 2012; Gidófalvi and Dong, 2012].

Modelling the spatial information of mobility with graphs is common in different analysis tasks such as location prediction, location inference, traffic anomaly detection, travelling time estimation and the detection of the most popular route [Zheng, 2015]. The nodes in these graphs are often intersections or important locations (e.g. home or office), and the edges are road segments with travelling times as the weights. However, there are different ways to define the graph. In [Yuan et al., 2012], the nodes are defined as regions and two nodes are connected with an edge if there is a certain minimum number of commutes between them. Chen et al. extract the turning points from raw trajectory data and, after clustering, they construct a graph that identifies the travelling probability to find the most popular route [Chen et al., 2011]. Zheng et al. construct a bipartite graph including users and locations for travel recommendations. In the graph, an edge between a user node and a location node exists if the user has visited the location [Zheng et al., 2009].

2.7 Spatio-temporal aspects of human mobility mining

Finally, some research explores both spatial and temporal aspects of human mobility [Scellato et al., 2011]. The problem of human location prediction falls into this area of research as both time and location are considered. Some location prediction methods predict the location of the user after a specific time, Δ, which is specified in each study. Δ could be from 10 minutes [Do and Gatica-Perez, 2014] to a couple of hours [Song et al., 2010] or even a year [Sadilek and Krumm, 2012]. In some studies, the effect of Δ on performance is investigated [Scellato et al., 2011; Jeung et al., 2010]. Do et al. changed Δ to predict a set of locations visited by the user [Do et al., 2015]. Furthermore, any other methods that process GPS data captured from users' smartphones are in fact looking at the spatio-temporal aspects of human mobility [Zheng et al., 2009] [Bastani et al., 2011].

2.8 Conclusion

Due to the vast application of understating human mobility in a variety of domains such as urban planning [Zheng et al., 2014], traffic management [Kitamura et al., 2000] and the spread of mobile [Kleinberg, 2007] and biological [Colizza et al., 2007] viruses, it has received considerable attention [Gonzalez et al., 2008]. Mining mobility patterns also benefits the users of mobile devices as it enables the mobile devices to offer context-aware assistance and information [Zhuang et al., 2011].

Mining human mobility requires dealing with diverse types of data, which results in addressing a wide range of problems ranging from segmentation to prediction. This chapter provides the necessary background on human mobility mining and pervasive signals. In a similar structure to the thesis, we discuss human mobility mining based on three aspects: temporal, spatial, spatio-temporal. We start with different types of pervasive signals and we then discuss human mobility problems while dealing with each type of data.

Chapter 3

Extracting transition times in daily activities

3.1 Introduction

This chapter of the thesis is concerned with finding transition times when one changes his/her activity during a specific period of time. The activity could be a fine-grained activity (e.g., sitting, standing or running) or a coarse-grained activity, which is an aggregate of low-level activities and has a more complex semantic (e.g., shopping, working at the office or commuting). In this chapter, we define fine-grained activities as low-level activities, and coarse-grained activities as high-level activities. Human daily mobility pattern is related to high-level activities because the user's location is affected by the user's high-level activities.

Studying the transition times of human daily activities is important not only for understanding the mobility patterns but also for providing context-aware services in pervasive systems [Roy et al., 2010]. For example, a user may want to get the news whenever he arrives home or before commuting to work. Furthermore, extracting transition times from daily activities is very useful in urban computing because it shows when people usually change their locations during a day. City planners use this information to improve transportation, urban planning, and environment [Zheng et al., 2014].

In this chapter, we present an information gain-based temporal segmentation method applicable to a wide range of pervasive data. Temporal segmentation approaches split time series into several homogeneous and non-overlapping intervals (segments). The output of applying temporal segmentation to the human activity data is a set of transition times (See

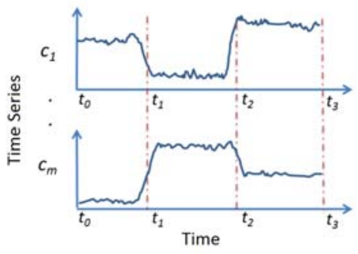


Figure 3.1: Temporal Segmentation with m *time series* $[c_1 \dots c_m]$ *and 3 segments.* t_1 *and* t_2 *are the transition times and* $[t_0\text{-}t_1]$, $[t_1\text{-}t_2]$ *and* $[t_2\text{-}t_3]$ *are the segments.*

Fig. 3.1).

To introduce our approach, Information Gain-based Temporal Segmentation (IGTS), let us consider smartphone connectivity data. Connectivity data specifies that the smartphone is connected to which access point or cell tower at each time. In this case, the results illustrate when users usually change their locations during a day. We treat the mean values of the time series (i.e. connectivity to the Cell Tower IDs) in each segment as a random variable and thus each segment has an entropy. Based on information theory, entropy reflects uncertainty. Correspondingly, in our case, entropy refers to the predictability of user locations. The IGTS algorithm finds low-entropy segments that means the users' locations are 'predictable'. As information gain gives the expected reduction in entropy after the segmentation, it is used as an effective cost function for finding coherent segments. As a result, the best segmentation is the one with the highest information gain value.

To find the best segmentation with the highest information gain, an optimization method should be deployed. We propose two approaches for optimization: TopDown and Dynamic Programming (DP). TopDown optimization is faster while it cannot guarantee to provide the global optima. On the other hand, DP optimization results in the global optima. In order to apply DP to our approach, the cost function is modified otherwise DP and the cost function are not compatible. Both TopDown and dynamic programming approaches are useful for specific applications due to the trade-off between the accuracy and running time.

In the next section, related work in temporal segmentation is discussed. We formally state the problem in Section 3.3. In Section 3.4, we present IGTS as a new temporal segmentation method that can be used for human activity data. The experiment results of our algorithm are reported in Section 3.5. Finally, Section 3.6 concludes our work.

3.2 Related Work

In this section, we briefly review the related work on temporal segmentation, the estimation of the number of segments, and transitions times in human activities.

3.2.1 Temporal segmentation

Time series segmentation approaches can be divided into three main groups: dynamic programming, heuristic, and probabilistic approaches [Panagiotou, 2015]. Dynamic Programming (DP) research dates back to 1950s and it has been used in many different contexts such as waveforms [Jackson et al., 2005] [Pavlidis and Horowitz, 1974], DNA sequence [Braun et al., 2000], and piecewise linear segmentation [Bellman, 1961]. The main idea of dynamic programming is that a complex problem is divided into small problems that are solved first. The results of the subproblems are saved into a table of results and used to solve the main problem. In temporal segmentation problem, dynamic programming is used as an optimization method in conjunction with a cost function. The basic dynamic programming method for segmentation is called k-segmentation that minimizes the variance inside the segments [Himberg et al., 2001] [Hiisila, 2007]. Kehagias et al. used DP algorithm for minimization of Hubert's segmentation cost to segment hydrological and environmental time series [Kehagias et al., 2006]. Guo et al. deploy dynamic programming and Schwarz's Bayesian information criterion for segmentation [Guo et al., 2015]. However, not all types of the cost functions are compatible with dynamic programming [Gionis and Mannila, 2005]. The total complexity of k-segmentation is $O(kn^3)$ that consists of the complexity of the optimization approach (i.e. DP) and the cost function (i.e. variance). However, using some reprocessing steps, it is improved to $O(kn^2)$ [Gionis and Mannila, 2005]. Nevertheless, it is still impossible to use k-segmentation for large datasets.

Heuristic approaches can be divided into three groups: sliding window, TopDown, and BottomUp approaches [Keogh and Kasetty, 2003] [Gensler et al., 2013]. In sliding window approaches, a window slides over the time series and a new segment is started when a specified error criterion is met. Online change-point detection approaches are similar to sliding window approaches but the window size is not fixed. These methods only consider local boundaries and thus are not able to provide a global model. Specifically, the transition times are chosen based on the whole time series not just a window of the time series. Although online methods do not need to estimate k, other parameters such as thresholds should be set that indirectly affects k [Banos et al., 2014]. If the thresholds for the change is set too low, the number of segments becomes too many. On the other hand, a high threshold results in a small number of segments. TopDown approaches start with one segment. Then, the time series is recursively partitioned until a specified error criterion is met in each step [Cheng et al., 2015; Yuan et al., 2013]. Cheng et al. also apply TopDown algorithm as an optimization method and use mutual information as a cost function for co-clustering on a co-occurrence matrix. They consider the table as a joint probability distribution of two discrete random variables for clustering in text analysis domain [Cheng et al., 2015]. In contrast to TopDown approaches, BottomUp approaches start with the maximum number of segments and the segments are merged until a specified error criterion is met.

Probabilistic-based segmentation algorithms consider the distribution of the data and transition times. For example, Bayesian techniques assume that the data originates from a Bayesian distribution and find the number and location of segments [Adams and MacKay, 2007]. Another example of probabilistic-based approaches is Hidden Markov Models (HMM) that assigns each segment to a state in the HMM. A change-point is alarmed when switching from one state to another [Mori et al., 2005]. Kawahara et al. deploy the ratio of probability densities instead of the probability densities themselves [Kawahara and Sugiyama, 2012]. Probabilistic methods can lead to acceptable results if the predefined model is correct. Matteson et al. do not make any assumptions about the distribution[Matteson and James, 2014]. The non-parametric estimation of the number of change-points is based on the divisive hierarchical algorithm included in the *ecp* package in the R statistical software. This uses divergence measure to detect any distribution change within an independent sequence. They deploy hierarchical significance testing to determine the statistical significance of change-points used as a stopping criterion for the iterative estimation procedure. Generally, Probabilistic-based methods require high computational time [Panagiotou, 2015].

There have also been proposed other methods in addition to the three aforementioned main categories. For example, Keogh et al. use a hybrid approach called SWAB (Sliding Window And BottomUp) to combine the advantages of Sliding Window and BottomUp algorithms [Keogh et al., 2001b]. Some approaches combine gradient decent and TopDown algorithm to escape local optima [Himberg et al., 2001]. The evolutionary method proposed by Chung et al. searches for a set of pattern templates which is determined by the user [Chung et al., 2002]. Yu et al. apply a cost function that depends on a covariance matrix using a low-complexity Pruned Exact Linear Time (PELT) method [Yu et al., 2014].

Similar to our method, E-Clustering introduced in [Yuan et al., 2013] applies the concept of entropy and information gain for road traffic segmentation across a day to find when the traffic changes. However, our proposed method is different in several aspects. First, the way we use entropy is different because we calculate the entropy considering the distribution of the average of the time series in each segment. On the other hand, in [Yuan et al., 2013], the entropy is calculated considering the distribution of members of the clusters extracted from the previous stage (i.e. V-Clustering). Second, E-clustering is not applicable to continuous time series. Third, E-clustering is not able to handle positively correlated data. Specifically, if the clusters' members reduce by a ratio at the same time, the entropy does not change and thus it is not possible to use the entropy calculation proposed by E-Clustering for the segmentation of heterogeneous activity data. On the other hand, our method is modified to handle both negatively and positively correlated time series.

3.2.2 k estimation

Estimating the number of segments/clusters, k, is also a challenging problem in segmentation/clustering. Typically, five common approaches are: cross-validation [Smyth, 2000] [Smyth, 1996], penalized likelihood estimation, permutation tests [Vasko and Toivonen, 2002], resampling [Roth et al., 2002], and knee point detection in evaluation metric graph [Salvador and Chan]. Probabilistic approaches use penalized likelihood estimation such as Bayesian Information Criterion (BIC) [Fraley and Raftery, 1998], Akaike Information Criterion (AIC), Minimum Message Length (MML) [Baxter and Oliver, 1996] and Minimum Description Length (MDL) [Hansen and Yu, 2001]. When there is an evaluation metric for the number of segments, the knee point in "number of clusters vs. evaluation metric" graph reveals k. In our case, information gain of each split performs as the evaluation metric. Therefore, the knee point reveals the best k because with the larger k there is no sharp increase in information gain. Various methods are used to find the knee point of the graph such as the largest magnitude difference between two points, the first data point with the second derivative above some threshold value [Foss and Zaïane, 2002], and the data point with the largest second derivative [Scott Harris et al., 2000]. While these methods consider local trends in the graph, other methods such as L-method [Salvador and Chan] try to find a point on the curve that is farthest from lines fitted to the entire curve.

3.3 Problem Definition

The heterogeneous time series in this chapter are sequences $\mathbf{X}$, which consist of n observations over various m-dimensional heterogeneous channels, $\mathbf{X} = \langle \mathbf{x}_1, \cdots, \mathbf{x}_i, \cdots, \mathbf{x}_n \rangle$, where $\mathbf{x}_i \in$

$\mathbb{R}^m$ denotes the i-th observation over m different channels. From the matrix perspective, $\mathbf{X}$ denotes an $m \times n$ matrix where x_{ji} denotes the i-th observation over j-th time series ($x_{ji} > 0$). In this matrix, column i is x_i and presents i-th observation while row j is c_j and presents all observations over the j-th time series.

The segmentation $\langle \mathbf{s}_0, ..., \mathbf{s}_k \rangle$ of the data set $\mathbf{X}$ consists of k non-overlapping segments $\mathbf{s}_j$ by k transition times $t_0 < \cdots < t_k < t_{k+1}$, where $t_0 = 1$ and $t_{k+1} = n$. Each observation belongs to exactly one continuous segment $\mathbf{s}_j = \langle \mathbf{x}_{t_j}, ..., \mathbf{x}_{t_{j+1}-1} \rangle$ and $\mathbf{T} = \langle \mathbf{t}_0, ..., \mathbf{t}_k \rangle$ is the set of transition times. $\mathcal{L} : N^k \rightarrow R$ is the cost function and the problem is finding a set of transition times that maximizes the cost function.

From the application point of view, we deploy temporal segmentation to detect transition times in human activities. In other words, temporal segmentation algorithms help us to capture human activity from multivariate time series. We aim to deal with the following problem:

Problem: Given time series data from multiple heterogeneous channels, how to detect times when the user changes his activities (e.g from "walking" to "running" or from "dining" to "commuting").

3.4 Temporal Segmentation based on Information Gain (IGTS)

In this section, we propose the Information Gain-based Temporal Segmentation (IGTS) method. After providing background on the concept of entropy, we introduce a novel information gain-based cost function. Then, we propose two optimization strategies to find the best set of transition times, and finally, we discuss a heuristic algorithm to find the best k.

3.4.1 Background: High/Low entropy variables

In information theory, Shannon entropy (H) measures the variance of a Probability Distribution Function (PDF) to show the information about the quantity of interests. When the variance of a PDF is high, little information can be inferred about the quantity of interests.

$$H = -\sum_{i=1}^{m} p(i) \log p(i) \tag{3.1}$$

where $p(i)$ denotes the probability of i^{th} outcome and H denotes the entropy of the probability distribution.

Many researchers have deployed the concept of entropy to measure the predictability in

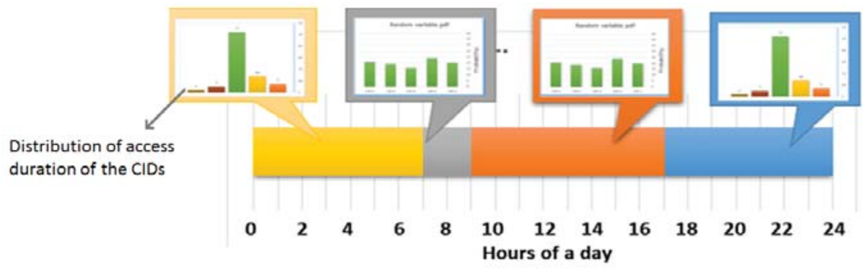


Figure 3.2: Segmentation of a day based on the user connectivity data. In this segmentation, the time intervals are [12am, 7am], [7am, 9am], [9am, 5pm], and [5pm, 12am]. Between 7am and 5pm, the user's location is less predictable.

human's behavior [Pham et al.] [McInerney et al., 2013]. Eagle et al. use the entropy of the cell towers (BTS antennas) connectivity distribution to measure the amount of predictable structure in an individual's life [Eagle and Pentland, 2006]. The predictability of the user's life can be quantified using entropy which is typically measured in bits. People who have entropic lives tend to be more variable and harder to predict. In contrast, low-entropy lives are characterized by strong patterns across all time scales.

The connectivity of the smartphone to the cell towers of the phone network has a PDF and entropy. Fig. 3.2 shows access duration distributions for four intervals for a specific user. The user has a high-entropy access duration between 7 am and 5 pm because the smartphone can be connected to all of the CIDs with the same probability. Therefore, it is hard to predict to which cell tower ID (CID) the user's phone is connected. In contrast, the access distribution to the CIDs has low entropy during midnight because the user is likely to connect to the dominant CID in this time period.

The main idea of the proposed method is that we try to find the segments with the lowest entropy so the user locations are predictable within each segment. Assume that the input time series is the CID connectivity data. In this case, each segment has an access distribution (as shown in Fig. 3.2). After calculating information gain for each segmentation, we find the best one with the highest information gain value. In fact, information gain performs as a cost function in this method. For the optimization method, we present two strategies that have their own pros and cons. Afterward, we propose a formula to identify the best candidate for the number of segments, k. Finally, we show that dealing with heterogeneous data, the positive correlation between the time series should be removed first.

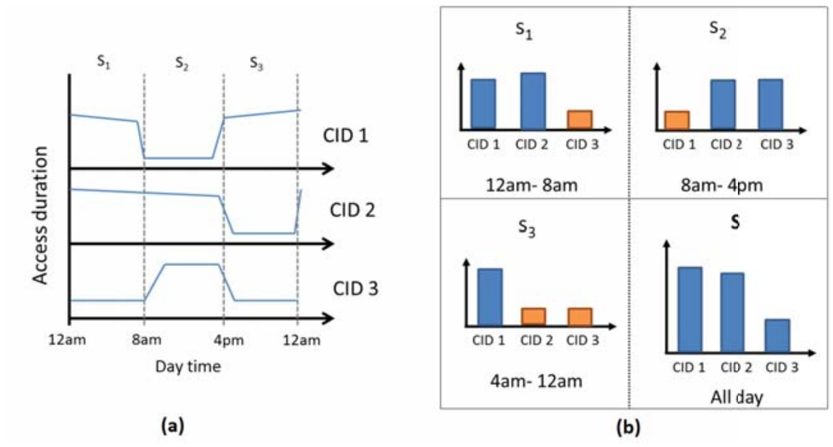


Figure 3.3: (a) Average access duration to the CIDs for a sample user (b) Each time interval has its own distribution and S presents the total distribution over a day.

3.4.2 Information gain-based cost function

Given a segmentation, we propose a cost function $\mathcal{L}$ by deploying the concept of Information Gain, which is the expected reduction in entropy caused by splitting the time series for a given segmentation [Shannon and Weaver, 2015]. Specifically, $\mathcal{L}$ is the expected reduction in the entropy caused by splitting $\mathbf{S}$ further, and is defined as follows:

$$\mathcal{L} = H(\mathbf{S}) - \sum_{i=0}^{k} \frac{|\mathbf{s}_i|}{|\mathbf{S}|} H(\mathbf{s}_i), \tag{3.2}$$

where k is the number of segments and $|\mathbf{s}_i|$ is the length of the i-th segment. $\mathbf{S}$ is the entire time series as a segment and $H(\mathbf{S})$ is the entropy of the whole time series. In our example, s_i is the CID access distribution over the i^{th} segment (see Fig. 3.3).

To calculate the entropy of the segments, the CID access distributions are considered as random variables. Therefore, in Eq. 3.1, $p(i)$ is the probability that user's smartphone is connected to the i^{th} CID. The following equation is used to calculate the entropy of the j^{th} segment:

$$H(\mathbf{s}_j) = -\sum_{i=1}^{m} p_{ji} \log p_{ji}, \tag{3.3}$$

$$p_{ji} = \frac{\sum_{q=t_{j-1}}^{t_j} c_{i_q}}{\sum_{p=1}^{m} \sum_{q=t_{j-1}}^{t_j} c_{p_q}} \tag{3.4}$$

where m denotes the number of CIDs (or time series), p_{ij} is the probability that the smartphone is connected to the i^{th} CID in the j^{th} segment, and c_{i_q} denotes the q^{th} observation of the i^{th} channel (CID).

To speed up the algorithm, we take advantage of the cumulative sum (or integral function) of c_i. In this case, first, the cumulative sum of all the time series should be computed to be used in each entropy calculation.

$$p_{ji} = \frac{\sum_{q=t_{j-1}}^{t_j} c_{i_q}}{\sum_{p=1}^{m} \sum_{q=t_{j-1}}^{t_j} c_{p_q}} = \frac{F_i(t_j) - F_i(t_{j-1})}{\sum_{p=1}^{m} F_p(t_j) - F_p(t_{j-1})}, \tag{3.5}$$

$$F_i(t) = \sum_{j=1}^{t} c_{i_j}, \tag{3.6}$$

F_i is the cumulative sum of c_i. This modification makes the algorithm much faster because there is no need to sum up the observations for each entropy calculation. Specifically, the complexity of calculating cost function $\mathcal{L}$ is reduced from $O(mn)$ to $O(m)$.

3.4.3 Optimization method

In addition to defining a cost function, an optimization method is needed to find the best segmentation, which is a NP-hard problem [Kleinberg et al., 2004] [Gionis et al., 2004]. Considering the complexity of full search, it is clearly impossible to be used for real-world data. We propose two optimization methods: TopDown and dynamic programming. TopDown optimization is faster while it cannot guarantee to provide the global optima. On the other hand, dynamic programming optimization is able to detect global optima but in a slower manner. As there is a trade-off between time and accuracy, either of the methods could be useful in different applications.

TopDown based optimization

A well-known and fast optimization method is the TopDown or binary-split approach that runs in a hierarchical and greedy manner. This is initialized by treating all observations as one segment. Then, in each step, one transition time is added without making any changes in the transition times it has once set. The split that reduces the total cost most is chosen in each step. The pseudo code of the recursive implementation is shown in Algorithm 3.1. The IG-Seg(S, k) returns the transition times by adding one to the output of IG-Seg($S, k-1$).

TopDown optimization is fast with the complexity of $O(nk)$. However, it may result in local optima.

Algorithm 3.1: IGTS and TopDown optimization

```
1  FUNCTION: IGTS-TopDown($S$,$k$) /* TopDown optimization; recursive
     implementation                                                    */
   Input: $S, k$
2  $-S$:Set of $m$ time series with the length of $n$
3  $-k$:Number of transition times
   Output: $T$
4  $-T$:Set of transition times
5  if $k == 0$ then
6      return $\phi$; /* terminating case of the recursive function    */
7  $T_{k-1}$ = IGTS-TopDown($S$,$k$-1);
8  $IG_{Max} = 0$; $t_k = 0$;
9  for $i = 1$ to $n$ do
10     if $IG(S, T_{k-1} \cup i) >= IG_{Max}$ then
11         $IG_{Max} = IG(S, T_{k-1} \cup i)$; /* IG(S,T) calculates the information gain
             of the segmentation                                        */
12         $t_k = i$; /* $t_k$ is the $k^{th}$ transition time          */
13 $T = T_{k-1} \cup t_k$;
14 return $T$ ;
```

Dynamic programming based optimization

Dynamic programming is a popular optimization strategy in the field of temporal segmentation [Hiisila, 2007] [Guo et al., 2015]. The main idea of dynamic programming is to divide a complex problem into small problems that are solved first and their results are saved to solve the whole problem. However, information gain is not compatible with dynamic programming because it is not a separable cost function for each segment.

Here, we define a variant version of $\mathcal{L}$ to fit the dynamic programming framework:

$$Max(\mathcal{L}) = Max(H(\mathbf{S}) - \sum_{i=0}^{k} \frac{|\mathbf{s}_i|}{|\mathbf{S}|} H(\mathbf{s}_i)) = H(\mathbf{S}) - Min(\sum_{i=0}^{k} \frac{|\mathbf{s}_i|}{|\mathbf{S}|} H(\mathbf{s}_i)) \tag{3.7}$$

This means that instead of maximizing the information gain, we minimize $\sum_{i=0}^{k} \frac{|\mathbf{s}_i|}{|\mathbf{S}|} H(\mathbf{s}_i)$ called weighted entropy (W_H) because $H(\mathbf{S})$ is a constant and it is the same for all segmen-

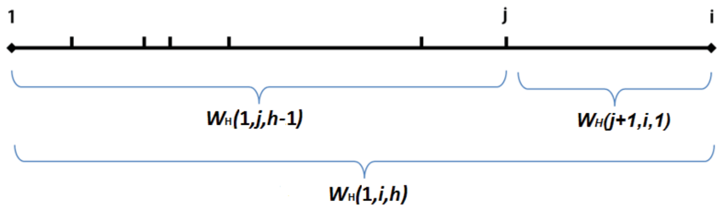


Figure 3.4: To compute $W_H(1, i, h)$, the best possible point for the last change-point, j, is investigated. The best j is the one that maximizes $W_H(1, j, h-1) + W_H(j+1, i, 1)$.

tation. Now, we can apply dynamic programming optimization to find a segmentation with minimum weighted entropy.

Dynamic programming-based IGTS is shown in Algorithm 3.2 and Fig. 3.4. $W_H(j, i, h)$ indicates minimum weighted entropy with h segments when the input time series is $\langle \mathbf{x}_j, \cdots, \mathbf{x}_i \rangle$. Clearly, we are looking for $W_H(1, n, k)$. In the first part of the algorithm, $W_H(j, i, 1)$ is calculated for all $1 \leqslant j < i \leqslant n$. In the second part, $W_H(1, i, h)$ is calculated recursively for all $1 < i \leqslant n$ and $1 < h \leqslant k$. To find $W_H(1, i, h)$, all possible situations for the $(h-1)$-th transition time are examined and the one that minimizes the weighted entropy is selected. The computational complexity of the optimization method based on dynamic programming is of order $O(kn^2)$.

3.4.4 k estimation

Estimating the number of the clusters/segments is still an open problem in clustering/segmentation. Some methods ignore providing a framework for identifying k although k is usually unknown before segmentation in real-world applications. For example, in routine discovery problems, k could be different from one user to another or from one day to another and depends on the mobility pattern of the user. In this section, we discuss how to choose the best k in a given range.

In our case, we use information gain as an evaluation metric and the knee point in number of segments vs. evaluation metric graph reveals k. The knee point, which is the point of maximum curvature of the graph, reveals the best k because with the larger k there is no remarkable increase in information gain.

Depending on the application, various methods can be used to find the knee point of the graph such as the largest magnitude difference between two points, the first data point with the second derivative above some threshold value [Foss and Zaïane, 2002], and the data point

Algorithm 3.2: IGTS and Dynamic Programming optimization

```
1  FUNCTION: IGTS-DP($S$,$k$)
   Input: $S, k$
2  $-S$:Set of $m$ time series with the length of $n$
3  $-k$: Number of transition times
   Output: $T$
4  $-T$:Set of transition times
   /* First part                                                   */
5  for $i = 1$ to $n$ do
6      for $j = 1$ to $i$ do
7          $W_H(j, i, 1) = \frac{|\mathbf{s}_1|}{|\mathbf{S}|} H(\mathbf{s}_1) = \frac{i-j}{n} H(\mathbf{s}_1)$;
   /* Second part                                                  */
8  for $h = 2$ to $k$ do
9      for $i = 1$ to $n$ do
10         $W_H(1, i, h) = W_H(1, i, h-1)$;
11         for $j = 1$ to $i-1$ do
12             if $W_H(1, j, h-1) + W_H(j+1, i, 1) \leqslant W_H(1, i, h)$ then
13                 $W_H(1, i, h) = W_H(1, j, h-1) + W_H(j+1, i, 1)$;
14                 $P(i, h) = j$;
                   /* The position of the last transition time is stored in
                   $P(i, h)$.                                              */
15 $T = \phi$;
16 $NextT = n$;
17 for $h$=0 to $k$-2 do
18     $NextT = P(NextT, k-h)$;
19     $T = T \cup NextT$;
20 return $T$ ;
```

with the largest second derivative [Scott Harris et al., 2000]. These methods consider local trends in the graph. On the other hand, non-local methods are more complex and analyse the entire graph. For example, L-method [Salvador and Chan] finds a point on the curve that is farthest from lines fitted to the entire curve. Local methods work well for smooth and monotonically increasing or decreasing curves. Here, we first prove that "number of segments vs. information gain" is a monotonic and non-decreasing graph. Then, we propose a local method to determine the optimal k.

Lemma: Information gain is non-negative.

Proof: Proof is based on Jensen's inequality [Peajcariaac and Tong, 1992] and details can be found here[1]. ■

Theorem: Maximum $\mathcal{L}$ is a monotonic non-decreasing function of k.

$$\mathcal{L}_k \leqslant \mathcal{L}_{k+1} \tag{3.8}$$

where $\mathcal{L}_k$ denotes the maximum information gain with k segments.

Proof: Let $T_k = (t_1, t_2, \ldots, t_k), 0 < t_1 < \cdots < t_k < n$ be the transition times of the segmentation with maximum information gain consisting of $k+1$ segments and $\mathcal{L}_k$ be the information gain of this segmentation. We prove that $T_{k+1} = (t_1, t_2, \ldots, t_k, t_{k+1}), t_k < t_{k+1} < n$ results in higher information gain than T_k. Based on T_{k+1} and T_k, we have

$$S_i^k = \begin{cases} S_i^{k+1}, \ 1 \leq i \leq k \\ \\ S_i^{k+1} \cup S_{i+1}^{k+1}, \ i = k+1 \end{cases} \tag{3.9}$$

where S_i^k is the i-th segment when the transition times are T_k. In fact, we add one transition time and split the last segment into two segments.

In this proof, we show that the segmentation caused by T_{k+1} has more information gain than segmentation caused by T_k. Consequently, the best segmentation with $k+1$ transition

[1]https://www.cs.cmu.edu/ ggordon/780-fall07/fall06/homework/15780f06-hw4sol.pdf

times has more information gain than the best segmentation with k transition times.

$$\begin{aligned}
&\mathcal{L}_{k+1} - \mathcal{L}_k \geq \\
&\left(H(S) - \sum_{i=1}^{k+2} \frac{|S_i^{k+1}|}{|S|} H(S_i^{k+1})\right) - \left(H(S) - \sum_{i=1}^{k+1} \frac{|S_i^{k}|}{|S|} H(S_i^{k})\right) = \\
&- \frac{|S_{k+1}^{k+1}|}{|S|} H(S_{k+1}^{k+1}) - \frac{|S_{k+2}^{k+1}|}{|S|} H(S_{k+2}^{k+1}) + \frac{|S_{k+1}^{k}|}{|S|} H(S_{k+1}^{k}) = \\
&\frac{|S_{k+1}^{k}|}{|S|}\left(H(S_{k+1}^{k}) - \frac{|S_{k+1}^{k+1}|}{|S_{k+1}^{k}|} H(S_{k+1}^{k+1}) - \frac{|S_{k+2}^{k+1}|}{|S_{k+1}^{k}|} H(S_{k+2}^{k+1})\right) = \\
&\frac{|S_{k+1}^{k}|}{|S|} IG_{tk} \geq 0
\end{aligned} \tag{3.10}$$

where IG_{tk} is the information gain caused by adding t_{k+1}. This inequality results from the lemma and information gain formula.■

The above theorem clarifies that there is an increase in information gain with each split. Therefore, the number of segments vs. information gain is a monotonic, non-decreasing and local knee point detection approaches can be applied to our problem. In our study, we found that none of the previous knee point detection methods gives a satisfactory result with our algorithm. We define the ratio ρ between the current and next increase in $\mathcal{L}$ to reveal the relationship between the number of segments and $\mathcal{L}$:

$$\rho_i = \frac{\mathcal{L}_i - \mathcal{L}_{i-1}}{\mathcal{L}_{i+1} - \mathcal{L}_i} \tag{3.11}$$

The best k among the given range is the one that has the largest ρ because the deviation of the trend has a sharp decrease. Fig. 3.5(b) shows an experiment on Human Activity Dataset (HAD), illustrating the effectiveness of ρ. The maximum value of ρ shows the best k is 5, which is equivalent to ground truth. In the experiments, we show that this formula works significantly better than other knee point detection formulas.

The proposed method can be applied to high-level activities and low-level activities. Therefore, to extract transition times without any information about k, we should provide a range in the search space to inform the model which one we are looking for. For example, if we want to find transitions in one day of data, we should specify that we are interested in transitions in high-level activities or transitions in low-level activities. We apply IGTS-TopDown to a day of data from Daily Life Routine Dataset that contains both high-level and low-level activity labels. The range of k is set between 1 and 150. Eq. 3.11 identifies the ρ.

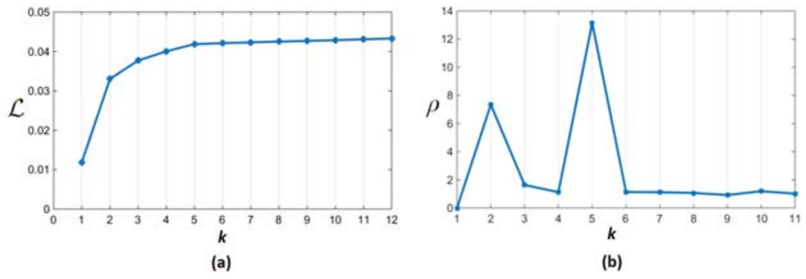


Figure 3.5: (a) $\mathcal{L}$ from segmentation for different k. The actual k is 5 in this example. (b) The k with the maximum ρ is selected as the number of segments.

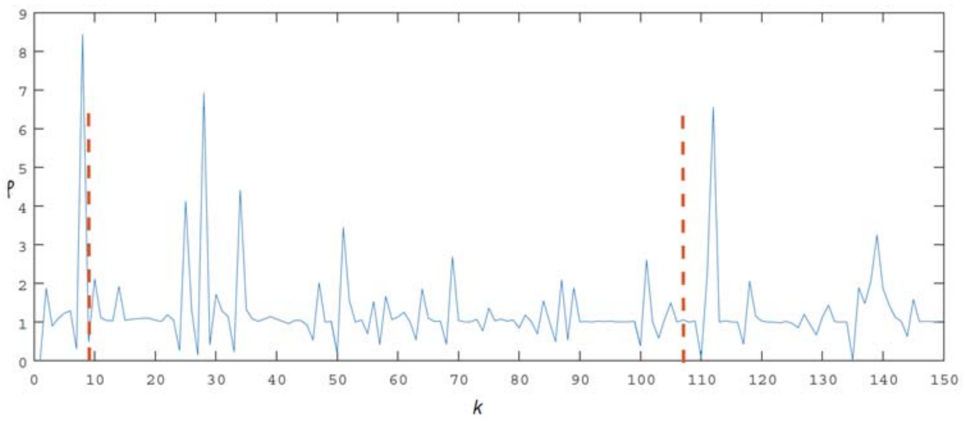


Figure 3.6: There are two peaks around the two red lines that shows the number of high- and low- level activities.

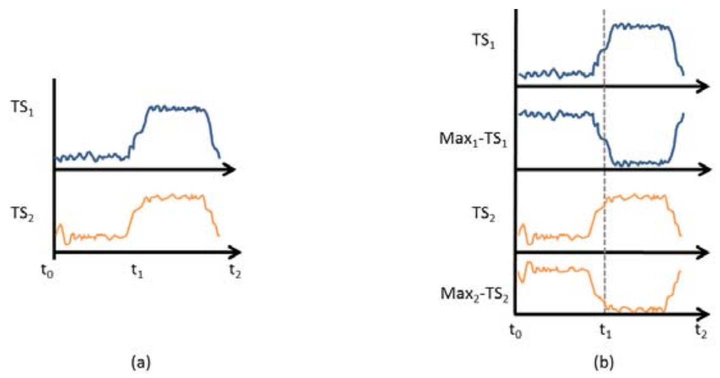


Figure 3.7: (a) Two time series with positive correlation. Transition times cannot be detected by simply applying information gain as a cost function because $[t_0, t_2]$, $[t_0, t_1]$, and $[t_1, t_2]$ have the same distribution. (b) Two complement time series are added. Thus, the time series are not positively correlated any more.

The high value of ρ shows a sharp increase in information gain that reveals the appropriate candidate for k. In this dataset, the number of high-level activities and low-level activities are 9 and 108 respectively. Fig. 3.6 shows the ρ for each k. The red lines are ground truth and knee point formula is calculated for each split. The proposed method for finding the number of segments works because there are two peaks around the actual number of segments for both high- and low- level activities.

3.4.5 Low-level activity and heterogeneous data

Here, we discuss how IGTS works with heterogeneous and positively correlated data sources, e.g. the time series from various sensor data. A positive correlation between two time series exists when by decreasing one of them, the other one also decreases and vice versa. The time series of CID access distributions do not have positive correlation because when the access to one CID increases in an interval, the access to the others decreases. On the other hand, the channels of sensor data, such as 3-axial linear acceleration and angular velocity, are highly correlated. Fig. 3.7(a) illustrates two time series with positive correlation, where the distribution is the same in $[t_0, t_1]$ and $[t_1, t_2]$. Simple information theory-based temporal segmentation methods are unable to detect the transition times in this case because the entropy is constant in all the segments. Thus, possible transition points could not be identified by using information gain as a cost function.

Assume X is an $m \times n$ matrix that presents the input time series where x_{ij} denotes the j-th observation over i-th time series (channel). c_i is the i-th row and denotes all observations over the i-th channel. To handle data heterogeneity problem, $\mathbf{X}$ is transformed as follows:

- Normalize $\mathbf{c}_i \in \mathbf{X}$ with $\sum_{j=1}^{n}(x_{ij} - \min_{\mathbf{c}_i})$:

$$\mathbf{c}_i \leftarrow \frac{\mathbf{c}_i - \min_{\mathbf{c}_i}}{\sum_{j=1}^{n}(x_{ij} - \min_{\mathbf{c}_i})}, \tag{3.12}$$

  where $\min_{\mathbf{c}_i}$ is the minimum of $\mathbf{c}_i$. This will remove the scale effects in heterogeneous data and give all the observations in different channels the equal priority. Without normalization, the algorithm relies only on the certain channels and leaves the other time series.

- Remove positive correlation by adding the negative value to the maximum value of each time series. Equation 3.13 shows the new matrix.

$$\begin{bmatrix} x_{11} & \dots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \dots & x_{mn} \\ max_{c1} - x_{11} & \dots & max_{c1} - x_{1n} \\ \vdots & & \vdots \\ max_{cm} - x_{m1} & \dots & max_{cm} - x_{mn} \end{bmatrix} \tag{3.13}$$

  where $\max_{\mathbf{c}_i}$ denotes the maximum of $\mathbf{c}_i$. After this stage, the number of rows is doubled and the input matrix becomes a $2m \times n$ matrix. The time series are not positively correlated anymore because when one time series falls, the corresponding negative one rises. Consequently, information theory metrics, e.g. information gain, can effectively distinguish the transition times.

Fig. 3.7(b) illustrates $\mathbf{X}$ transformed from time series in Fig. 3.7(a). Experiment results show that considering the negative time series not only makes the method more generic but also improves the accuracy. Furthermore, without this transformation, it is not possible to analyse a single time series because the entropy is always zero for one channel.

3.5 Experiments and Evaluation

In this section, we examine the proposed IGTS method on a variety of different data sources, including sensor data [Zhang and Sawchuk, 2012], device-free RFID data [Yao et al., 2015],

Table 3.1: Semantic of the datasets

Dataset	**Meaning of a transition time**	**Meaning of a segment**	**Example of segments**
Device-Free posture recognition [Yao et al., 2015]	change in the posture	a posture is performed	standing freely, standing straightly, sitting, sitting leaning back, sitting leaning left, sitting leaning right, sitting leaning forward, lying in bed, etc
USC Human Activity Dataset [Zhang and Sawchuk, 2012]	change in the low-level activity	a low-level activity is performed	walking forward, walking upstairs, walking downstairs, running forward, jumping, sitting, standing, sleeping, etc
Movement detection by Bluetooth iBeacons	movement of the device collecting Bluetooth RSSI	The device is still	different locations of the device
Daily Life Routine Dataset [Huynh et al., 2008]	change in the high-level activity	a high-level activity is performed	commuting, office work, lunch, dinner
Device analyser connectivity data [Wagner et al., 2014]	usual change in location during a day	The smartphone is connected to a certain set of CIDs	working hours when the smartphone is connected to the CIDs near the workplace

connectivity data [Wagner et al., 2014], and mobility data [Huynh et al., 2008]. The proposed method has no limitation on activities. In the experiment, our main focus is to demonstrate the generality of the method on various datasets collected with different types of devices and for various purposes. Table 3.1 shows the semantics of each dataset. We compare IGTS with different segmentation methods. All the datasets contain multivariate time series either from correlated or from uncorrelated channels. USC-HAD [Zhang and Sawchuk, 2012] and Device Analyser [Wagner et al., 2014] are heterogeneous. USC-HAD consists of time series from accelerometer and gyroscope, and device analyser consists of different connectivity data, such as Cell Tower IDs and Wi-Fi access points. We apply the proposed method to an unlabelled connectivity dataset and measure the inner cluster distance. Finally, we show the performance of the proposed method for finding the actual k.

3.5.1 Datasets

Synthetic data (Syn): It contains 2-4 time series with the length of 420. The transition times are selected randomly with the uniform distribution. The value of the time series in each segment is set to a random integer between 1 to 6. To make the data more realistic, we add white Gaussian noise with the signal to noise ratio (SNR) of 10dB. In this case, the transition times can hardly be recognized by eye. Fig. 3.8(b) illustrates a sample synthetic time series.

Device-free posture recognition by RFID (RFID): The data is collected from an array of 9 passive RFID tags to recognize 12 different human postures [Yao et al., 2015]. Each posture is performed by 6 subjects for 60 seconds. The data collection is device-free. The tags are placed on a wall and the subject performs predefined postures between the wall and the RFID antenna. The corresponding sequence of RSSI is collected at the sampling rate of 2Hz.

USC Human Activity Dataset (HAD): The dataset includes the most basic and common low-level human activities in daily life from a diverse group of human subjects [Zhang and Sawchuk, 2012]. In total, it contains 12 activities collected from 14 subjects while each consists of 1000 to 10000 samples. The 12 human activities include: walk forward, walk left, walk right, go upstairs, go downstairs, run forward, jump up and down, sit and fidget, stand, sleep, elevator up, and elevator down. To make the data more real, each subject was asked to perform 5 trials for each activity on different days. The data is captured by a 3-axis accelerometer and a 3-axis gyroscope, sampled at 100 Hz (See Fig. 3.8(b)).

Movement detection by Bluetooth iBeacons (iBeacon): iBeacon hardware transmitters are a class of Bluetooth low energy devices that broadcast their locations to nearby portable electronic devices such as smartphones and tablets. We use a smartphone to collect the data receiving signal strength indicator (RSSI) of four Bluetooth iBeacons. During the data collection, a smartphone with fixed position starts collecting RSSI of the iBeacons. After 1-2 minutes, the position of the smartphone is changed within the experiment area and the change time is recorded. Over the experiment, 5326 RSSI samples are collected from iBeacons at the frequency of 2.5 Hz. We apply temporal segmentation to this dataset to find the transition times when the position of the smartphone is changed.

Daily Life Routine Dataset: This dataset contains not only 34 daily low-level activity labels but also 4 high-level routine class annotations such as commuting, lunch, office work, and dinner [Huynh et al., 2008]. Two wearable accelerometer-based sensors were placed at the dominant wrist and in the right pocket of a single male subject as he was performing his everyday activities. The accelerometer sampling rate data is 100Hz, and the features (i.e., mean and variance of acceleration of each axis) are computed over a 0.4-second window.

Smartphone logs from Device Analyser: Device Analyser gathers data about running background processes, wireless connectivity, GSM communication, and some system

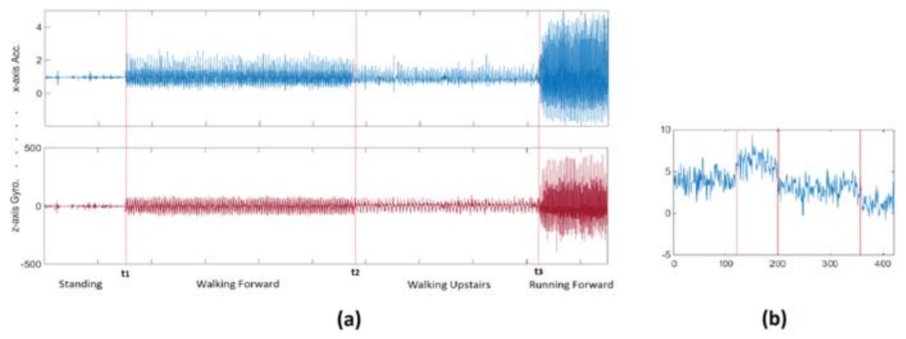


Figure 3.8: (a) Sample multivariate time series from USC Human Activity dataset (b) Sample synthetic time series

status and parameters. For privacy purposes, MAC addresses, WiFi SSIDs, CIDs, and other forms of identification are hashed. Therefore, there is no ground truth or information about the subjects and semantic of the location [Wagner et al., 2014]. In our experiments, the time series are CIDs access distributions. In our experiment, the proposed algorithm is applied to 27789 days of CID connectivity data that is collected from 271 devices. Each device has at least two weeks of data.

3.5.2 Labelled low-level activity datasets

We evaluate different versions of our algorithm on low-level activity datasets while k is known. Furthermore, we examine the proposed k estimation method.

Evaluation metrics and experiment setting

The detected transition times are evaluated from two aspects: *i*) FN (False Negative) is the number of missed transition times *ii*) closeness of the detected transition times to the actual transition times. For the first aspect, the precision of the detection is reported. A detected transition time is considered as true positive if the closest actual transition time is closer than 10% of the length of the total time series. For the second aspect, Root Mean Square Error (RMSE) is calculated among the true positive detected transition times. Thus, it is always below 10%.

For k estimation method, the output of our proposed method is compared with the actual k. For each k, we run the algorithm 40 times on different synthetic time series. The results include confusion matrix as well as the accuracy over 320 runs.

The experiments run on a desktop PC with the configuration of Intel(R) Core i7, 3.4GHZ and 8G RAM.

Baseline and proposed methods

For comparison, we use k-segmentation, which is a foundation of segmentation algorithms in many works [Hiisila, 2007]. It is based on dynamic programming and finds the segmentation with the minimum variance inside the segments. The second baseline is a state-of-the-art method proposed by Matteson et al. in which the positions of the change-points are estimated based on hierarchical clustering and measuring the differences in multivariate distributions [Matteson and James, 2014]. This method is included in the *ecp* package in the R statistical software.

We evaluate the three versions of our algorithm. IGTS-TopDown and IGTS-DP are presented in Algorithm 3.1 and Algorithm 3.2, respectively. We also evaluate IG-Based, which is a simple version of our algorithm without normalization, transformation, and speed-up stages. The difference between IGTS-TopDown and IG-based is that in IGTS-TopDown, the time series are doubled and normalized. Furthermore, the cumulative sum or integral of the time series is used to speed up the algorithm. IG-Based is similar to E-Clustering [Yuan et al., 2013] in terms of the cost function and optimization method. However, E-Clustering is not applicable to continue and multivariate time series because it takes advantage of information gain in a different way.

For k estimation method, we find 10 transition times and the corresponding information gains. Then, we compare our method with 4 knee point detection formulas [Foss and Zaïane, 2002], including:

- $\mathcal{L}_i - \mathcal{L}_{i-1}$: difference between magnitudes;
- $2\mathcal{L}_i - \mathcal{L}_{i-1} - \mathcal{L}_{i+1}$: second derivative;
- $\frac{\mathcal{L}_i}{\mathcal{L}_{i-1}} - \frac{\mathcal{L}_{i+1}}{\mathcal{L}_i}$: deference between the ratios;
- $\frac{\mathcal{L}_i - \mathcal{L}_{i-1}}{\mathcal{L}_{i-1}}$: relative difference between magnitudes.

Results

Table 3.2 shows the experiment results for 4 datasets including one synthetic and three real-world datasets. The dataset name, the number of segments, the number of time series, and the average length of the time series are specified in each row. The performance and the running time are the average values over 20 runs. The only exception is k-segmentation on USC-HAD in which the number of runs was reduced to four because it was too time-consuming. In experiments with the precision value less than %50, the MSE is not mentioned because it is non-meaningful.

Four issues can be inferred from the results. First, IGTS methods are faster than the baselines. k-segmentation is very slow specially for large n. The method proposed in [Matteson and James, 2014] is slow on USC-HAD dataset because it takes a long time to estimate the distribution of the heterogeneous time series. Second, IG-based works as well as IGTS-DP and IGTS-TopDown on the iBeacons and RFID datasets because the time series in these datasets have negative correlations. Increasing the RSSIs of some iBeacons leads to decreasing the other RSSI because when the smartphone gets closer to some iBeacons, it becomes farther from others. Third, neither IG-based nor k-segmentation can capture the transition times in USC-HAD that contains heterogeneous data from accelerometer and gyroscope. Fourth, IGTS-DP is slower than IGTS-TopDown but it provides better results. The output of IGTS-DP and IGTS-TopDown are different if TopDown approach results in local optima. Otherwise, both methods find the global optima and the results are the same. The probability that TopDown approach ends up in local optima or global optima depends on the size and type of the time series. HAD time series are long and IGTS-TopDown cannot usually find the global optima, which is the best answer. That is why IGTS-DP performs better than IGTS-TopDown on HAD dataset.

Fig. 3.9 shows the confusion matrix of the proposed k estimation formula. Each row of the confusion matrix represents the instances in a predicted class while each column represents the instances in an actual class. The entry in the row i and column j denotes how many times our approach detected k as i while the actual number is j. As we run the experiment 50 times for each k, the sum of each column is 50. The results show that the proposed approach can find the correct k.

The accuracy of our k estimation method and other knee point detection formulas are reported in table 3.3. Obviously, for IGTS, the proposed formula works significantly better than other knee point detection formulas.

Table 3.2: Experiment Results for low-level activity

Input				{Matteson 2014}			k-segmentation			IG-Based TopDown			IGTS TopDown			IGTS DP		
Data	k	m	n	P*	T*	R*	P	T	R	P	T	R	P	T	R	P	T	R
Syn	6	2	420	100	0.1	0.1	100	2.4	0.1	76	1.0	5.9	100	0.1	0.1	100	1.4	0.1
	7	2	420	100	0.1	0.1	100	2.7	0.1	76	1.1	8.5	100	0.1	0.1	100	1.4	0.1
	6	3	420	100	0.1	0.1	100	2.6	0.1	80	1.2	8.4	100	0.2	0.1	100	1.5	0.1
	5	5	420	100	0.1	0.1	100	2.8	0.1	67	1.2	8.0	100	0.2	0.1	100	1.5	0.1
RFID	3	9	748	100	0.1	1.1	100	10.2	1.1	100	0.1	1.3	100	0.1	1.1	100	3.4	1.1
	7	9	1496	100	0.8	1.1	100	80.8	1.2	100	0.9	1.0	100	1.0	1.0	100	17.6	1.0
	11	9	2244	100	2.4	1.0	100	430.7	0.9	100	3.4	1.1	100	3.7	1.1	100	38.1	1.1
iBeacon	3	4	790	100	0.1	1.1	100	0.1	1.7	100	0.1	1.5	100	0.1	1.6	100	0.1	0.9
	4	4	988	100	0.1	0.8	97	0.1	2.2	100	0.1	2.0	100	0.1	1.8	100	0.1	1.4
	5	4	1185	100	0.1	1.7	97	0.2	1.7	100	0.1	2.1	100	0.1	1.8	100	0.1	1.3
HAD	2	6	12600	95	21.8	0.2	25	804.6	-	19	130.0	-	95	0.3	0.5	98	4.2	0.1
	3	6	16800	93	132.8	0.4	50	2766.8	-	16	539.5	-	98	0.3	0.6	100	19.1	0.1
	4	6	21000	93	782.9	0.3	33	4498.2	-	17	811.5	-	98	0.4	0.5	98	35.9	0.1

* P (%): Precision, T (s): Running Time, R(%): Root Mean Square Error (RMSE)

Table 3.3: Evaluation on different knee point detection approaches to find k

Method	$\mathcal{L}_i - \mathcal{L}_{i-1}$	$2\mathcal{L}_i - \mathcal{L}_{i-1} - \mathcal{L}_{i+1}$	$\frac{\mathcal{L}_i}{\mathcal{L}_{i-1}} - \frac{\mathcal{L}_{i+1}}{\mathcal{L}_i}$	$\frac{\mathcal{L}_i - \mathcal{L}_{i-1}}{\mathcal{L}_{i-1}}$	$\frac{\mathcal{L}_i - \mathcal{L}_{i-1}}{\mathcal{L}_{i+1} - \mathcal{L}_i}$
Description	magnitude difference	second derivative	ratio difference	relative magnitude difference	proposed formula
Accuracy	10%	32%	25%	22%	**71%**

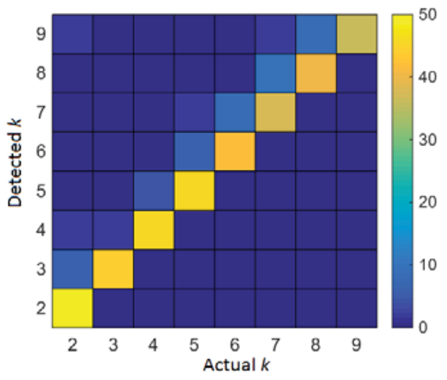


Figure 3.9: Confusion matrix of k estimation approach

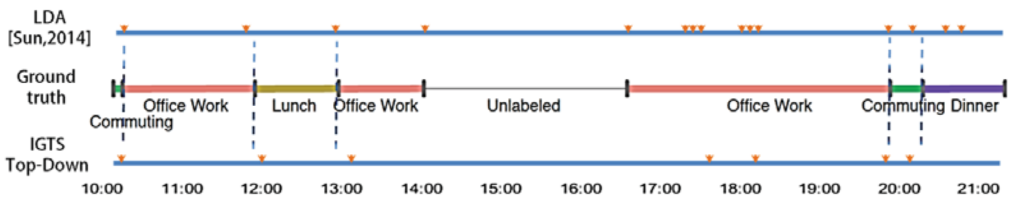


Figure 3.10: Qualitative comparison between the proposed method and Latent Dirichlet Allocation (LDA) [Sun et al., 2014]

3.5.3 Labelled high-level activity datasets

In this experiment, we apply IGTS-TopDown and IGTS-DP to Daily Life Routine dataset to extract transition times in high-level activities. The high-level activities in this dataset are commuting, lunch, office work, and dinner.

Baseline method

For comparison, we choose the method proposed in [Sun et al., 2014]. They use Latent Dirichlet Allocation (LDA) to infer high-level activities and report qualitative evaluation.

Results

IGTS-TopDown and IGTS-DP have the same results on Daily Life Routine dataset. The qualitative evaluation of our method as well as LDA is shown in Fig. 3.10.

It is obvious that our approach has less false positive errors. Using the proposed heuristic approach, detected k is 7 while LDA approach finds 15 transition times. The two false positives of our approach are around 6 pm. By investigating the ground truth labels, we found that “walking freely” is frequent between 5 pm and 6 pm. Furthermore, LDA is costly because it has two complex stages (see related work). On the other hand, the total running time for segmentation and finding the best k is only 10 seconds for our algorithm.

If we continue the segmentation, next transition times show the transitions in low-level activities. IGTS algorithm, first, extracts the transitions in the coarse-grained activities, and then finds the transitions in the fine-grained activities. We define coarse-grained activities as high-level activities, and fine-grained activities as low-level activities. Specifically, the earlier transition times denote the transition times in high-level activities. By continuing segmentation, the segments become shorter and later transition times denote transitions in low-level activities. As discussed, by choosing the range of k, we can specify whether we are

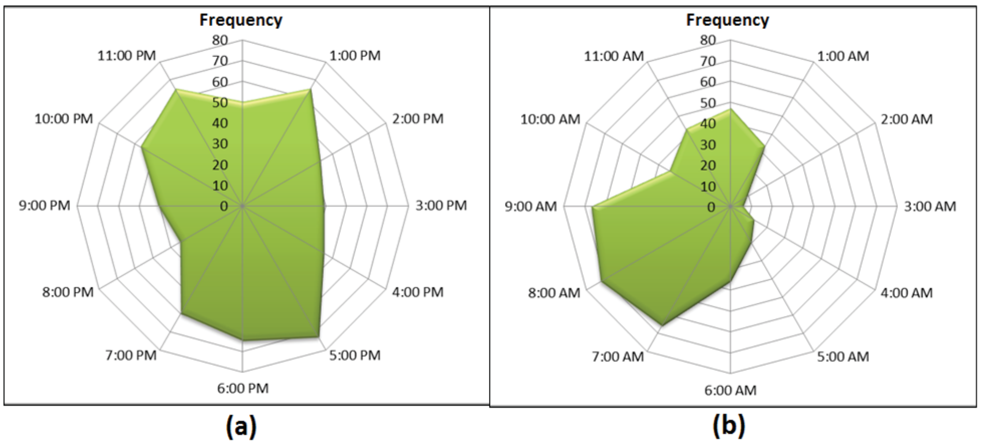


Figure 3.11: (a) Transition times in midnight and morning. (b) Transition times in afternoon and night. The number of transition times for all users is reported hourly. It can be inferred that users mainly change their places during 8am and 5pm.

interested in low-level or high-level activities. For example, in figure 6, if the range is between 2 and 10, the best k is 8, which indicates the number of segments for high-level activities.

3.5.4 Segmentation at the routine level from unlabelled dataset

Device Analyser dataset provides the connectivity data for each user and identifies the connected CID at each time. By processing connectivity data, we segment 24 hours of a day into several parts. The output of segmentation shows when the user usually changes his location during a day. In this experiment, the accuracy is 15 minutes which means the transition times have only four possible formats: hh:00 , hh:15, hh:30, and hh:45, where hh indicates the hour of the day. IGTS-TopDown processes the whole data which is 33 Gigabytes in just 4 hours.

Results

Fig. 3.11 shows the frequency distribution of the transition times for all users per hour. For example, 65 users usually change their locations between 1 pm and 2 pm. According to the graphs, users are less likely to change their locations during nights. In contrast, 5 pm in the evening and 8 am in the morning are the times when the users have the most movements. This information can help city planners to improve transportation, urban planning, and environment [Zheng et al., 2014].

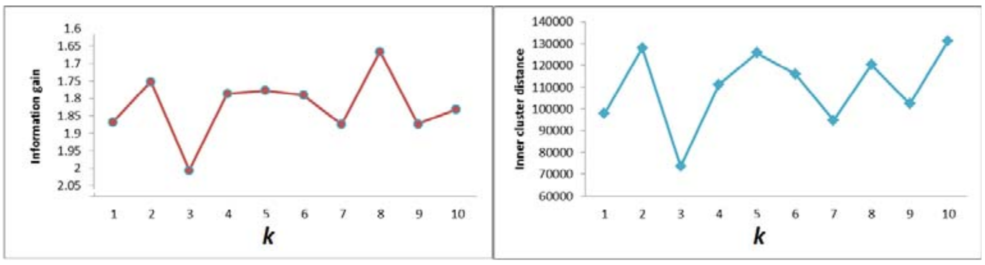


Figure 3.12: Comparison between proposed approach and inner cluster distance.

Validation

Because of the lack of ground truth in Device Analyser dataset, we examine our method by computing inner cluster distance. We show that our algorithm tends to find coherent segments that means the connected CID does not change so much within the segments. There have been several ways for measuring coherence of the segments. When a segmentation is evaluated based on the segments themselves, it is called inner cluster distance. The experiment illustrates that segmentation with the higher information gain value results in lower inner cluster value.

To calculate inner cluster distance, the distance between each couple of instances is calculated for each segment. In the CID example, each instance is a 15-minute slot and the distance between two instances is calculated based on the difference in access distributions. For example, [10:00, 10:15] and [11:45, 12:00] have a lower distance value if their user's smartphone connects to similar CID during these periods. Finally, the average over all such distances indicates the inner cluster distance. To compare the outcome of our algorithm with the inner cluster distances, first 10 different segmentations and 10 users are chosen randomly. Then, information gain and inner cluster distance are calculated for each segmentation and each smartphone data. The average of both information gain and inner cluster distance shown in Fig. 3.12. Considering the reverse order of information gain axis, it is clear that the segmentation with high information gain value has less inner cluster distance which means access durations to the CIDs have a little change within each interval. It should be noticed that measuring information gain is much faster than measuring the inner cluster distances because it does need to compare every two instances. In this experiment, it takes 478.9 ms to calculate the inner cluster distances while our method accomplishes in just 3.7 ms.

3.6 Conclusion

In this chapter, we present a new temporal segmentation method, IGTS, for extracting transition times in daily activities (e.g. when the user goes to work from home) that contain rich information for analyzing human mobility pattern. Although we introduce IGTS for mining human mobility data, it is quite generic and can be applied to other domains as well. We deploy the concept of entropy in our method. For each segment, the distribution of the average values of the time series is considered as random variables. The entropy of the distribution is calculated for each segment and then, information gain, which is the average reduction in entropy, is obtained for the segmentation. We argue that the best segmentation is the one with the highest information gain. To the best of our knowledge, none of the previous works has applied information gain in this way. Two processing stages are introduced to enable the proposed IGTS method to handle heterogeneous data. We also show that working with the cumulative sum (or integral function) of the time series instead of the original time series decreases the computational cost leading to a faster algorithm.

The proposed cost function is used in conjunction with two optimization methods: Top-Down and Dynamic Programming. The original concept of information gain is not applicable to dynamic programming, and we have introduced a modified cost function to be fit with dynamic programming optimization. Furthermore, we prove that information gain increases with each split and proposed a local knee point detection formula to find the number of segments. Experiments show that the proposed formula performs much better than the existing ones.

To evaluate IGTS in the domain of human mobility, we apply it to Daily Life Routine dataset containing 4 high-level routine class annotations such as commuting, lunch, office work, and dinner. The qualitative evaluation shows that our method has less false positive errors and outperforms LDA. We also apply our method to the connectivity data collected by Device Analyser software. In this case, the transition times denote the usual transitions in a day such as when the user goes to work. Integrating the result from 271 users, the most number of transitions happens at 5 pm and 8 am. In addition to human mobility data, IGTS is examined on a range of datasets including activity recognition data, connectivity data, and movement data from smartphones, wireless infrastructure, and device-free environment. The results reveal the effectiveness and robustness of the proposed approach.

Chapter 4

Movement graph summarisation

4.1 Introduction

The efficient modeling of user's mobility on a limited storage allows us to perform the mobility analysis algorithms on smartphones and other resource-limited devices. An intuitive approach is to model the user's movement with a movement graph. The graph-based mobility models realize graphs in which nodes are the locations visited by the user and edge weights denote the distances or traveling times between locations. It should be noted that the movement graph can be defined in different ways. A node may represent region covered by a WiFi access point, region covered by a cell tower ID (CIDs), region defined by geographic coordinates, suburb, point of interest (POI), stay pint, or road intersection. Similarly, each edges between two nodes can denote a relation between the nodes such as distances, traveling time, frequency of commutes.

Dealing with an original movement graph poses several challenges. First, the graph is usually large that cannot be fitted into the smartphone memory. In this case, the user's movement cannot be analysed on the smartphone device and need to be passed to the server, which reduces the reliability of the services. Second, due to the size of the graph, it is hard to visualize, understand and query the graph. Figure 4.1 (a) shows a sample movement graph of a student at RMIT University commuting between the library, shopping centre and campus. Figure 4.1 (b) shows the same graph summarised and easy to understand where the edge weights denote the number of the movements. Furthermore, when the graph is big the queries are processed slowly, especially when running on a resource-limited device such as a smartphone. These challenges can be addressed by summarising the movement graph to get a summarised graph where each node represents a cluster of nodes.

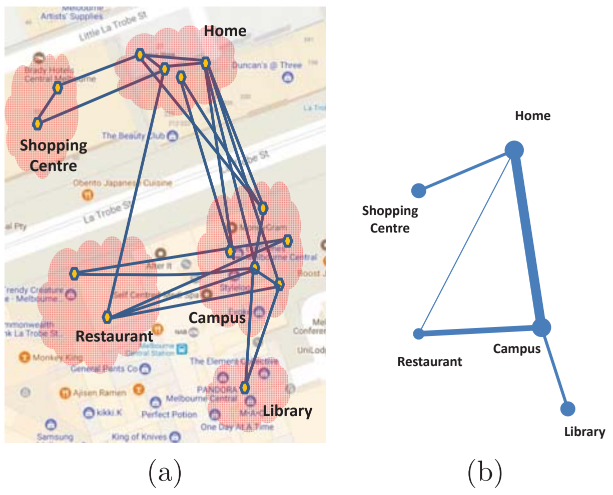


Figure 4.1: (a) Sample movement graph (Original graph). Each node is a stay point. (b) Summary of the same movement graph (Coarse graph). Each node represents a cluster of stay points.

In this chapter, we present *Shrink*, a summarisation method that reduces the size of the graph while preserving the distances between the nodes. In the summarised graph, known as the coarse graph, each node, called supernode, represents several nodes in the original graph. The edge weights in the coarse graph are defined in the way that the distances between the nodes are preserved. The coarse graph can be easily analysed and stored in lower space. Furthermore, as the complexity of distance-based queries such as shortest path queries is highly dependent on the size of the graph, *Shrink* improves the performance in terms of time and storage. Although Shrink is designed to summarise the movement graph, it is quite generic and can be applied to other types of graphs including road network, friendship network, collaboration network, web graph and social network.

Summarising the movement graph, we compute the changes in the distances caused by summarisation. Furthermore, we investigate its effect on two human mobility mining problems: location prediction and similarity mining. The location prediction algorithm on the coarse graph causes coarse-grain results. Regarding computing the similarity, summarisation reduces the computational cost but at the same time increases the uncertainty of the results. summarisation also addresses privacy issues as the exact location of the user cannot be determined in the coarse graph easily. Although in some situations, the user's location can be still inferred, the location prediction generally becomes harder after summarisation.

The main contributions of this chapter are as follows:

- We propose a compression method *Shrink* that has the least effect on the distances between the nodes.
- The proposed method is evaluated on not only movement graphs but also other real-world data sets, including road network, friendship network, collaboration network, web graph and social network, etc.
- We investigate the effect of summarisation on the performance of two human mobility mining problems.

In the next section, we provide background on graph summarisation. We formally state the problem in Section 4.4. In section 4.5, first, we present the baselines for defining the equations and discuss why the equations are suitable for assignment of the new weights. Then, an overview of *Shrink* is provided. Three stages of *Shrink* are described in section 4.6 and 4.7. We evaluate our method in terms of time, accuracy and storage in section 4.8. Finally, sections 4.2 and 4.9 discuss related work and conclusion, respectively.

4.2 Related work

Graph compression methods can be categorized into two groups: general compression and query-friendly compression also known as summarisation. General compression methods preserve the information of the entire graph and answer all types of queries. However, these methods highly depend on the graph type, coding mechanism, and extrinsic information. Furthermore, these methods need decompression before querying the graph [Chierichetti et al., 2009]. Specifically, the graph should be restored first to answer even simple queries. Some works has been studied graph compression for Web graph and social networks [Raghavan and Garcia-Molina, 2003] [Boldi et al., 2011] [Chierichetti et al., 2009]. Similar to our work, graph summarisation approaches target specific classes of queries, such as neighborhood [Maserrat and Pei, 2010] [Navlakha et al., 2008] [Brisaboa et al., 2014], reachability [Feder and Motwani, 1995] [Moyles and Thompson, 1969] [van Schaik and de Moor, 2011] [Fan et al., 2012], path [Buneman et al., 2003], pattern queries [Fan et al., 2012], connectivity [Zhou et al., 2010], and distance-based queries [Ruan et al., 2011] [Toivonen et al., 2011]. For example, Aho et al. reduced graphs by substituting a simple cycle for each strongly connected component to speed up reachability queries [Aho et al., 1972]. Brisaboa et al. focus on compression of web graphs to reduce the space requirements while running queries such as successors and predecessors extraction is possible on the compressed graph [Brisaboa et al., 2014]. Some researches address the similar problem as a graph simplification problem [Ruan et al., 2011] [Zhou et al., 2010] and graph summarisation [LeFevre and Terzi, 2010] [Čebirić et al., 2015] [Seah et al., 2012].

Shrink differs from the current compression methods in the following: (1) *Shrink* is developed for reachability and distance-based queries; (2) most of the methods are designed for unweighted graphs while *Shrink* can be applied to both unweighted and weighted graphs [Boldi et al., 2011] [Maserrat and Pei, 2010] [Fan et al., 2012]; (3) *Shrink* provides compressed data structure, the coarse graph, that can be directly queried without decompression [Chierichetti et al., 2009] [Navlakha et al., 2008]; (4) Some of the compression methods only reduce the number of the edges while *Shrink* reduce the number of nodes and edges [Feder and Motwani, 1995] [Moyles and Thompson, 1969] [van Schaik and de Moor, 2011]; (5) *Shrink* can be performed incrementally for temporal graphs while most of the above methods are batch algorithms, requiring to decompress the whole graph to do the change such as edge insertion or node removal [Ruan et al., 2011] [van Schaik and de Moor, 2011]. (6) *Shrink* not only specifies the distance between the two nodes but also provides the actual path containing laying nodes [Gubichev et al., 2010].

Fan et al. extract a coarse graph from a directed unweighted graph to speed up reachability queries [Fan et al., 2012]. *Shrink* has several advantages compared to Reachability Preserving Compression (RPC). First, RPC only preserves reachability while *Shrink* is advantageous for all types of distance-based queries. Second, *Shrink* compresses the graph much faster. The complexity of compression is linear in the number of nodes for *Shrink* while it is quadratic for RPC (i.e. $O(\sigma|V|^2)$ where σ denotes the average degree in the graph). Third, unlike RPC, *Shrink* grantees the reachability with any compression ratio. RPC has a limit for compression ratio to preserve the reachability ranging from 0.02% to 14.70%. Fourth, the error and compression ratio is flexible in *Shrink*. Therefore, it is possible to terminate the coarsening process at any time while having a semi-compressed graph.

Ruan et al. propose a Distance Preserving Graph Simplification (DPS) for unweighted graphs [Ruan et al., 2011]. In the first stage, some nodes are selected from the original graph and then the selected nodes are connected to each other with weighted edges. This method is designed to preserve the distance between every non-local pair ($\forall x, y\ d(x, y) \leq \epsilon$). The output is a weighted coarse graph that cannot be simplified again. DPS is costly and the largest examined graph contains less than 63,000 nodes. The compression ration is also fixed. Another distance preserving compression method is Compression of Weighted Graphs (CWG) which is compared with *Shrink* [Toivonen et al., 2011]. Table 4.1 summarises the

Table 4.1: Feature comparison between RPC [Fan et al., 2012], DPS [Ruan et al., 2011], CWG [Toivonen et al., 2011] and Shrink

Properties	RPC	DPS	CWG	*Shrink*
Target Graphs	Directed Unweighted	Undirected Unweighted	Undirected Un/Weighted	Undirected Un/Weighted
Query Type	Reachability	Distance-based	Distance-based	Distance-based
Adjustable CR	×	✓	✓	✓
Flexible Running Time	×	×	✓	✓
Provide the Path	×	×	×	✓
Compression works for	all pairs	non-local pairs	all pairs	all pairs
Required Parameters	Nonparametric	ϵ	λ	Nonparametric
Largest examined graph for distance-based queries	-	62K (nodes)	200K (nodes)	2.7M (nodes)

* σ: average degree in the graph

features of RPC, DPS, CWG, and *Shrink*.

4.3 Graph summarisation

Nowadays, it is increasingly common to find graphs with millions of nodes in various domains. Due to the commonness of large graphs, graph summarisation is becoming an important research topic. In this process, also known as graph simplification, the complexity of the graph is reduced while certain characteristics of the graph are preserved. Graph summarisation can be performed by reducing the number of edges, nodes or extracting a high-level abstraction of the graph. A similar problem is graph summarisation where summary graph uncovers the underlying topology of the original graph [LeFevre and Terzi, 2010] [Čebirić et al., 2015] [Seah et al., 2012] [Chan et al., 2013] [Leskovec and Faloutsos, 2006].

Graph summarisation has three main purposes. First, graph summarisation algorithms produce a simpler graph that can be queried faster than the original graph. This is useful for graph-based mining algorithms and also more complex problems (e.g. measuring similarities between graphs). Distance-based queries such as shortest-path have a great importance [Sommer, 2014] [Zhu et al., 2013] [Akiba et al., 2013] [Akiba et al., 2015] [Gao et al., 2011]. Many fundamental tasks in graph mining, such as computing diameter, closeness, centrality, and betweenness centrality, are dependent on computing shortest path distance. It has countless applications in transportation networks [Zhu et al., 2015] [Hong et al., 2017] [Abraham et al., 2011], networking [Althfer et al., 1993] [Wu et al., 2014], and databases [Schenkel et al., 2004]. Distance preserving graph summarisation speeds up the shortest path queries because they run on a smaller graph. Second, the compressed graph, also known as the coarse graph, can be stored in less space. Over the past few years, due to increasing the size of graph-structured databases, it becomes challenging and expensive to store the data, and graph summarisation techniques are deployed to reduce space consumption [LeFevre and Terzi, 2010] [Brisaboa et al., 2014] [Lohrey et al., 2013]. Third, graph summarisation algorithms help the users to understand and visualize the high-level structure of the graph [Aggarwal and Wang, 2010] [Hennessey et al., 2008] [Rafiei, 2005]. It is almost impossible to understand the information encoded in large graphs with thousands or even millions of nodes by only visual inspection [Tian et al., 2008]. The coarse graph, produced by compression, is smaller and easier to be visualized [Fjällström, 1998].

In this chapter, we introduce a novel distance preserving compression method *Shrink*, which can be used to query and store both weighted and unweighted graphs, e.g. enhancing shortest-path or other distance-based algorithms. Compressing with *Shrink* has the least effect on the distances between nodes. Specifically, when merging two nodes to a supernode, a system of equations is introduced to minimize the distance variations caused by this merge. The rationale behind this is to keep the mean of caused error equal to zero. These equations determine the edge weights connected to the supernode. After each merging, the number of nodes decreases by one and merging stage, called coarsening stage, continues until the desired size is achieved. The next stage is executing stage where the distance based query runs on the coarse graph. The last stage, refining stage, is optional that provides the path between the queried nodes. We have theoretically proved that for long paths, the error of compression converges to zero if merging errors are independent.

4.4 Problem formulation

In this section, we first introduce the necessary notation to describe the problem formulation, and then we state the problem.

Definition 1: *Original graph* is the input graph which is a triple $G = (V, E, w)$ where V is a set of nodes (or vertices), $E \subset V \times V$ denotes edges, and $w : E \to R^+$ assigns a non-negative weight to each edge $e \in E$.

Here, the original graphs can be either unweighted or weighted. For unweighted graphs, the same weight can be assigned to all edges. The notations used in this chapter are listed in Table 4.2.

Definition 2: *Coarse graph*, $G' = (V', E', w')$, is the compressed graph. $V' = \{v_1', ..., v_n'\}$ is a partition of V (i.e. $v_i' \subset V$ for all $i, \bigcup_i v_i' = V$, and $v_i' \bigcap v_j' = \emptyset$ for all $v_j \neq v_i$). Namely,

Table 4.2: Definition of the variables

Variable	Definition
G	Original graph: $G = (V, E, w)$
V	Set of the nodes in the original graph
E	Set of the edges in the original graph: $E \subset V \times V$
w	Weights on E in the original graph:$w : E \to R^+$
x, y	two nodes in the original graph $x \in V, y \in V$
u, v	To be merged nodes in the original graph $u \in V, v \in V$
G'	Coarse graph: $G' = (V', E', w')$
V'	Set of the nodes in the coarse graph
E'	Set of the edges in the coarse graph: $E' \subset V' \times V'$
w'	Weights on E' in the coarse graph: $w' : E' \to R^+$
v'	Supernode in the coarse graph $v' \in V'$
k	Number of v''s neighbours
$N(u)$	Set of u's neighbours that are not connected to v
$N(v)$	Set of v's neighbours that are not connected to u
$N(uv)$	Set of Common neighbours of u and v
N	Set of neighbours of u and v: $N = N(u) \cup N(v) \cup N(uv)$
v_i	A neighbour of u or v
w_{vi}	weight of the edge between v_i and v
w_{ui}	weight of the edge between v_i and u
w'_{vi}	weight of the edge between v_i and v'
$l(v_i, v_j)$	Length of the path that connects v_i and v_j and crosses u or v in the original graph
$l'(v_i, v_j)$	Length of the path that connects v_i and v_j and crosses v' in the coarse graph

each node ${v_i}' \in V'$, also known as a supernode, may consist of some nodes in G. E' denotes the edges set $E' \subset V' \times V'$, $w' : E' \to R^+$. In contrast to the nodes, there is no mapping between the edges.

$$E' = \{(u', v')|u \in u', v \in v', (u, v) \in E\} \tag{4.1}$$

Specifically, two supernodes are connected if and only if there is a node in one supernode that is connected to a node in the other supernode. Here, the main problem is assigning weights to the new edges. To this end, we define and solve equations to have new weights with the least effects on the distances between nodes.

Definition 3: The *distance* between x and y is $d(x, y)$, which is the length of the shortest path between x and y in the original graph. The shortest path is a path with the lowest total sum of edge weights. Similarly, $d'(x, y)$ denotes the length of the shortest path between the supernodes that contain x and y in the coarse graph.

Definition 4: Error of the compression for nodes x and y, $Err(x, y)$, is the difference between the distance of x and y in the original graph and the distance of the supernodes that x and y belongs to in the coarse graph.

$$Err(x, y) = \left|d(x, y) - d'(x, y)\right| \ , x \neq y \tag{4.2}$$

Definition 5: Normalizing $Err(x, y)$ with $d(x, y)$, we have the *relative error* of nodes x and y.

$$RErr(x, y) = \frac{|d(x, y) - d'(x, y)|}{d(x, y)} \ , x \neq y \tag{4.3}$$

where $RErr(x, y)$ denotes the relative error.

Definition 6: Given G and G', the compression ratio of the coarsening stage is defined as $CR(G') = \frac{|V'|}{|V|}$. The number of edges is not included in the definition.

Problem: Given the original graph G and a compression ratio CR, $0 < CR < 1$, how to define G' such that the sum of the errors is minimum over all pairs. Specifically, the cost function that should be minimized is as follow:

$$\sum_{x,y \in V} Err(x, y) \ , x \neq y \tag{4.4}$$

4.5 Shrink method

In this section, we introduce the overview of the proposed method. Then, we present preliminary concepts and discuss the rationale behind the method.

4.5.1 Overview

Shrink consists of three components: 1)Coarsening stage: This is the first stage in which the coarse graph is constructed by merging pairs of connected nodes iteratively until the desired size is achieved. The main challenge is finding the new edge weights for the supernodes with the least effect on the distances in the graph. We define a system of linear equations to find the new weights. We also propose a speed-up technique to solve the equations. It should be noted that the coarse graph is constructed once and after that, it can be queried for all types of the distance-based queries. 2) Executing stage: In this stage, the query is executed on the coarse graph. As the coarse graph is smaller than the original graph, execution is faster and requires lower heap size. 3) Refining stage: This stage is optional. If only an estimation of the distance is enough, the refining stage is not needed, and the original graph can be discarded. On the other hand, if the laying nodes on the path are needed, the refining stage is necessary. In the refining stage, we extract a subgraph by projecting the output path to the original graph. Then, the path is refined by rerunning the query on the subgraph from the original graph. Figure 4.2 demonstrates the flowchart of *Shrink* including the three stages.

Our main focus is on the coarsening stage where we try to compress the graph while preserving the distances. In the next subsection, we discuss our principal for assigning the new weights to edges connected to a supernode.

4.5.2 Preliminary

Merging a pair of nodes changes the distances because by merging, the structure of the graph is changed. The merging error is consequently defined in terms of the change in the length of a shortest-path caused by the merging.

Definition 7: Assume that v and u are merged into v' (See Figure 4.3). pa is a shortest-path between two unknown nodes in the original graph that passes through u or v. As a result, the length of pa will be affected by the merging. The merging error caused by merging u and v on pa ($M_{uv}(pa)$) is defined as follow:

$$M_{uv}(pa) = l(pa) - l'(pa) \tag{4.5}$$

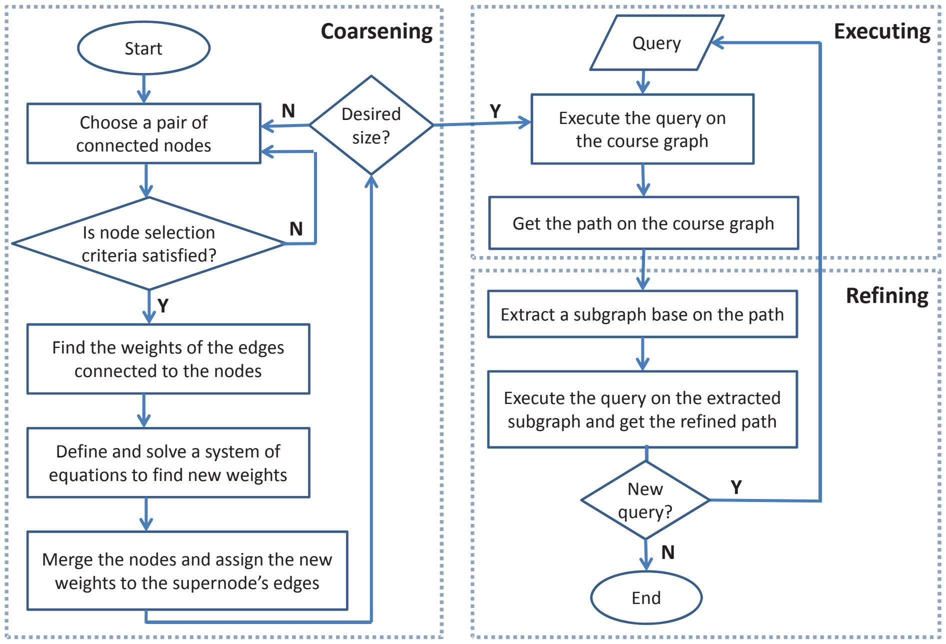

Coarsening
Start
Choose a pair of connected nodes
N
Desired size?
Is node selection criteria satisfied?
N
Y
Find the weights of the edges connected to the nodes
Define and solve a system of equations to find new weights
Merge the nodes and assign the new weights to the supernode's edges
Executing
Query
Y
Execute the query on the course graph
Get the path on the course graph
Refining
Extract a subgraph base on the path
Execute the query on the extracted subgraph and get the refined path
New query?
Y
N
End

Figure 4.2: Shrink *has three stages: Coarsening, Executing, Refining. Queries are run on the coarse graph produced in the coarsening stage.*

where $l(pa)$ is the length of the part of pa in the original graph that will be changed after merging. This part of pa consists of edges connected to u and v. $l'(pa)$ is the length of the part of pa in the coarse graph that is changed after merging. This part of pa consists of two edges connected to v'.

The value of the merging error depends on how the path passes through the merged nodes and their neighbors. For example, in Figure 4.3, if pa passes through v_i, u, v, and v_j, $l(pa)$ is $w_{ui} + w_{uv} + w_{vj}$. In the coarse graph, $l'(pa)$ is $w'_{vi} + w'_{vj}$. The merging error of u and v on this path is $(w_{ui} + w_{uv} + w_{vj}) - (w'_{vi} + w'_{vj})$. The merging error could be positive or negative depending on v_i, v_j, w'_{vi} and w'_{vj}.

By considering the occurrence of the neighbors of u and v in the shortest paths, we define a random variable regarding the merging error. To have a better understanding of this random variable, assume that someone starts marking the edges on the shortest paths one by one and we can only see u and v neighbours and the rest of the graph is a black box. The random variable presents the merging error when the edges connected to u or v are marked. Specifically, a probability assigned to the random variable, such as p_{ij}, denotes the probability that edges connected to v_i and v_j from the visible part are marked.

The PDF (Probability Density Function) of the random variable shows the impact of the merging on the distances/shortest paths. For example, from the PDF, we can calculate the probability that a random path passing through u or v becomes shorter after the merging. The PDF is computed based on the probability of occurrence of each pair of neighbors. Figure 4.4 illustrates a long path in the coarse graph consisting of many supernodes as well as the PDF of the merging errors. The total error, the difference between the length of the path in the coarse and original graphs, is equal to the sum of the merging errors of the supernodes laying on the path. The total error is also a random variable and has a PDF.

After each merging, new weights should be assigned to the supernode edges. *Our main principle for defining the new weights is to keep the mean of each merging error equal to zero.* Here, we investigate the effect of this principle on mean and variance of total error.

Mean of error

According to probability theory, the mean of the sum of some random variables is equal to the sum of the random variable means [Wackerly et al., 2007]. In our problem, we have

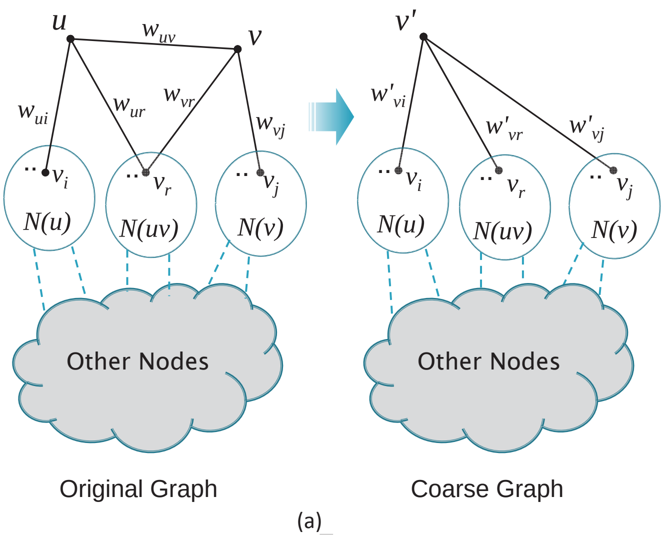
$$E(M(pa)) = E(M_{u_1v_1}(pa)) + ... + E(M_{u_mv_m}(pa)) \tag{4.6}$$

where pa is the shortest path between two random nodes and $M(pa)$ is the total error which is the difference between the length of pa in the original graph and the length of pa in the coarse graph. u_i and v_i is a merged node pair laying on pa. Therefore, the mean of the error of a path is zero regardless of the correlation between merging errors of the supernodes laying on the path because it is equal to the sum of the merging error means that are set to zero. To sum up, if we consider the shortest path between two random nodes, the compression has zero effect on the path length on average.

We also argue that when the average error is zero, Mean Square Error (MSE) is minimum. According to probability theory, if μ and σ^2 are the mean and variance of the error respectively, MSE is equal to $\mu^2 + \sigma^2$ [Wackerly et al., 2007]. Based on this theorem, if the variance of error is σ^2, the minimum MSE is when $\mu = 0$. This means our weight assignment, which leads to a zero-mean error, is the *best* one when the variance is fixed.

Variance of error

The average of a sufficiently large number of iterates of independent[1]zero-mean random variables converges to zero, regardless of the underlying distributions (Law of Large Numbers



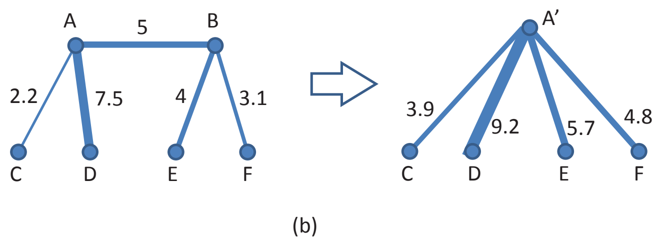


Figure 4.3: (a) u and v are merged into v' connected to the neighbors of both u and v. The connectivity pattern of the other parts of the graph remains the same. (b) A simple example where A and B are merged into A'. The weights of the new edges connected to A' are determined by Shrink*.*

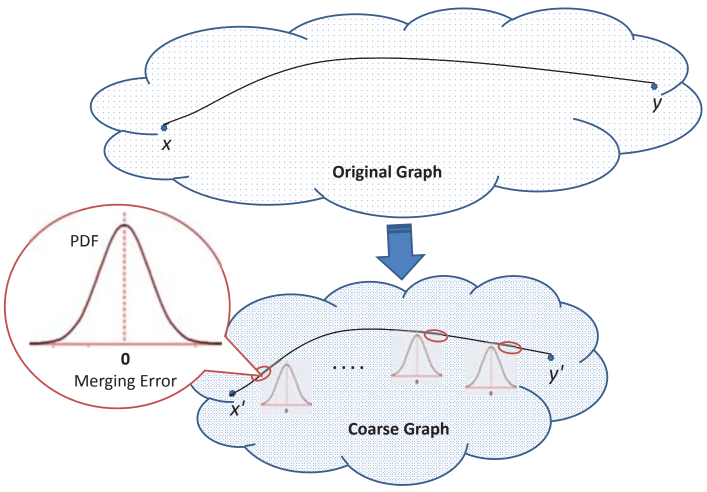


Figure 4.4: Each supernode on the path causes merging error that has a PDF. The proposed weight assignment makes every merging error mean equal to zero.

[Wackerly et al., 2007]). Merging errors are zero-mean. Based on Law of Large numbers, if they are independent, for long paths, the relative errors are zero and the distances are preserved. Intuitively, some supernodes increase the length of the path while others decrease the path length. Overall, the change is negligible because the increases and decreases compensate for each other.

$$\lim_{m\to\infty} \frac{E(M_{u_1v_1}(pa)) + ... + E(M_{u_mv_m}(pa))}{m} = \lim_{m\to\infty} \frac{E(M(pa))}{m} = 0 \tag{4.7}$$

$$\lim_{m\to\infty} \frac{E(M(pa))}{m \times \overline{l(pa)}} = RErr(pa) = 0 \tag{4.8}$$

where $\overline{l(pa)}$ is the average value of $l(pa)_{u_iv_i}$ over u_i and v_i. $RErr(pa)$ is the relative error of path pa.

To sum up, zero-mean merging error has two advantages. First, the total error of a path becomes a zero-mean random variable that has the least MSE. Second, for the long paths, the total error is nearly zero if merging errors are independent.

[1]Two random variables are independent when receiving information about one of the two does not change our assessment of the probability distribution of the other.

4.6 Coarsening stage

The coarsening stage is the main stage where the size of the graph is reduced. By considering the underlying idea, our method seeks to reduce the graph size by merging pairs of nodes one after another, setting though the mean of merging error to zero. Let u is the closest neighbor of v. It should be noted that u and v could be supernodes that result from the previous steps. The edge weights and the connectivity pattern of u and v are sufficient to define a system of equations and find the new weights.

Figure 4.3 (a) illustrates the general overview of the problem, focusing on the connectivity of nodes u and v. All of the nodes that are connected to either u or v are connected to supernode v' but with different weights. The neighbors of u and v can be divided into three sets: $N(u)$, $N(v)$, and $N(uv)$. Nodes in set $N(u)$ are connected to u but not to v. Set $N(uv)$ consists of the nodes that are connected to both u and v, while $N(v)$ is the mirror of $N(u)$ and consists of the nodes that are connected to v but not u. Suppose N is the set of the neighbors of u and v, $N = N(u) \cup N(v) \cup N(uv)$ and k is the total number of neighbors, $k = |N|$. w_{uv} denotes the weight between u and v. Figure 4.3 (b) shows a simple example in which A snd B are merged.

Given the edge weights in the original graph, the problem is finding k new edge weights between v' and the nodes in $N(u)$, $N(v)$, and $N(uv)$ so that the mean of the merging error becomes zero. For each merging, the equations should be defined and solved, and then new weights should be assigned to the new edges.

4.6.1 Defining equations

In this section, we define a system of k linear equations in the k variables to find the new weights where k is the total number of the neighbors of u and v. The equations imply that the changes in the length of paths that pass through the merged nodes after and before merging have a zero mean. We only consider parts of the paths that contains u or v and new edges because other parts do not change.

Each of the k equations is based on one of the new edge weights connecting v' to $v_i \in N$. The main principle states that the mean of the difference between the paths in the original and coarse graphs should be zero. Therefore, the mean (or average) value of path lengths that pass $v_i \in N$ and u or v in the original graph should be equal to the mean value of path lengths that pass v_i and v' in the coarse graph.

To calculate the mean value, the probabilities of the occurrence of the neighbors in the shortest paths are needed. Following the principle of indifference [Jaynes, 2003], it is assumed that all pairs of neighbors have the same probability of occurrence in the shortest paths because there is no information about the other parts of the graph. This assumption may not be true, and it drops accuracy since low weight edges are used more in shortest paths. Future work involves new techniques to estimate the usage probability distribution of the neighbors.

Lemma: If the sum of $l(v_i, v_j)$ and the sum of $l'(v_i, v_j)$ over v_j are the same, the merging error of u and v for the paths that pass v_i and v' is a zero-mean random variable.

Proof: We show that if the sum of $l(v_i, v_j)$ and the sum of $l'(v_i, v_j)$ are the same, then the mean of $l(v_i, v_j)$ and the mean of $l'(v_i, v_j)$ become the same. Therefore, the merging error becomes a zero-mean random variable because merging error is the difference between $l(v_i, v_j)$ and $l'(v_i, v_j)$.

$$\sum_{v_j \in N, j \neq i} l'(v_i, v_j) = \sum_{v_j \in N, j \neq i} l(v_i, v_j) \tag{4.9}$$

$$\sum_{v_j \in N, j \neq i} p \times l'(v_i, v_j) = \sum_{v_j \in N, j \neq i} p \times l(v_i, v_j) \tag{4.10}$$

$$E(M_{uv}(v_i)) = \sum_{v_j \in N, j \neq i} p \times l'(v_i, v_j) - \sum_{v_j \in N, j \neq i} p \times l(v_i, v_j) = 0 \tag{4.11}$$

where p is the probability that v_i and v_j occur on a shortest-path connecting two nodes of the graph. $l(v_i, v_j)$ indicates the length of the part of the paths that includes v_i, v_j, and the edges to be deleted in the original graph. Likewise, $l'(v_i, v_j)$ indicates the length of the part of the paths that includes v_i, v_j, and new edges in the coarse graph. $M_{uv}(v_i)$ is merging error of the shortest paths passing v_i and v'. In these equations, it is not necessary to consider the complete length of the paths since other parts of the graph remain unchanged. Equations 4.9-11 focuses on v_i and the sum in these equations is over v_j (v_i is fixed). This means we have one equation for each neighbour. $l'(v_i, v_j)$ is a function of variables to be determined such as w'_{vx}, $w'_{v'y}$, and $w'_{v'z}$ while $l(v_i, v_j)$ is a function of w_{ux}, w_{vy}, and w_{uz} which are known and constant. Equation 9 is a key equation that clarifies that sum of the part of the shortest paths' lengths that pass through v_i should be the same in the original and coarse graphs. In fact, if we want the merging process to have the minimum change in the shortest paths, the sum of $l'(v_i, v_j)$ and $l(v_i, v_j)$ should be the same. Based on the lemma, k linear equations are defined.

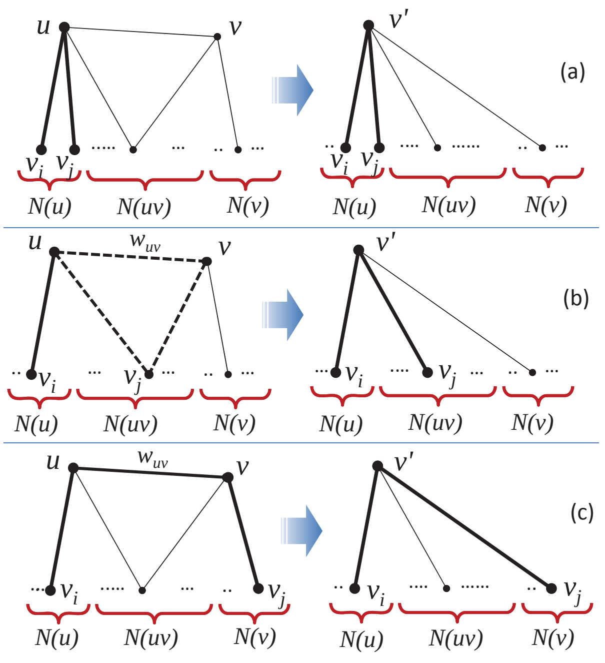


Figure 4.5: When v_i is in set $N(u)$, the shortest paths that crosses v_i and v_j can be categorized into three cases: (a) $v_j \in N(u)$, (b) $v_j \in N(uv)$, and (c) $v_j \in N(v)$. (See rows 1-3 in Table 4.3)

$$\begin{aligned}\sum_{j\neq i, v_j\in N(u)} l'(v_i,v_j) + \sum_{j\neq i, v_j\in N(v)} l'(v_i,v_j) + \sum_{j\neq i, v_j\in N(uv)} l'(v_i,v_j) = \\ \sum_{j\neq i, v_j\in N(u)} l(v_i,v_j) + \sum_{j\neq i, v_j\in N(v)} l(v_i,v_j) + \sum_{j\neq i, v_j\in N(uv)} l(v_i,v_j)\end{aligned} \tag{4.12}$$

Eq. 4.12 is the extension of Eq. 4.9. The left side of the equation contains variables that should be computed while the right side of the equation is constant. The constant part can be determined based on the given edge weights in the original graph. In the rest of this section, $l(v_i, v_j)$ and $l'(v_i, v_j)$ are computed based on whether v_i and v_j are in set $N(u)$, $N(v)$, or $N(uv)$.

Equations for $v_i \in N(u)$

In this subsection, we write an equation for a node $v_i \in N(u)$. $l'(v_i, v_j)$ and $l(v_i, v_j)$ are computed based on whether node v_j belongs to $N(u)$, $N(v)$ or $N(uv)$. Figure 4.5 shows three possible situations for v_j that is used to find the length of the part of the shortest paths which contains v_i and merged nodes. This path is shown by bold lines in the new and original graphs. Figure 4.5(a) demonstrates a situation in which v_j belongs to set $N(u)$. Therefore, we can determine the first part of the right and the left side of Eq. 4.12.

$$\begin{aligned}
&\sum_{v_j \neq v_i, v_j \in N(u)} l'(v_i, v_j) \\
=&\sum_{v_j \neq v_i, v_j \in N(u)} (w'_{vi} + w'_{vj}) \\
=&(|N(u)| - 1)w'_{vi} + \sum_{v_j \neq v_i, v_j \in N(u)} w'_{vj}
\end{aligned} \tag{4.13}$$

$$\sum_{v_j \neq v_i, v_j \in N(u)} l(v_i, v_j) = \sum_{v_j \neq v_i, v_j \in N(u)} (w_{ui} + w_{uj}) \tag{4.14}$$

In other words, if both i and j belong to $N(u)$, Eq. 4.13 and Eq. 4.14 indicate the sum of the length of the affected paths in the new and original graph, respectively. Figure 4.5(b) is more complex. The exact value of the $l(v_i, v_j)$ depends on the w_{uj}, w_{uv}, and w_{vj}. If $w_{uj} < w_{uv} + w_{vj}$ then $l(v_i, v_j)$ is w_{uj} otherwise it is $w_{uv} + w_{vj}$.

$$\sum_{v_j \in N(uv)} l'(v_i, v_j) = \sum_{v_j \in N(uv)} (w'_{vi} + w'_{vj}) = |N(uv)|w'_{vi} + \sum_{v_j \in N(uv)} w'_{vj} \tag{4.15}$$

$$\sum_{v_j \in N(uv)} l(v_i, v_j) = \sum_{v_j \in N(uv)} (w_{ui} + min(w_{uj}, w_{uv} + w_{vj})) \tag{4.16}$$

Finally, if v_j belongs to set $N(v)$, the path that includes v_i, u, v, and v_j could be part of a shortest-path whose length is $w_{ui} + w_{vj} + w_{uv}$. Nodes in set $N(uv)$ cannot be in this path because v is the closest neighbor of u and w_{uv} has the minimum weight among the neighbors of u.

$$\sum_{v_j \in N(v)} l'(v_i, v_j) = \sum_{v_j \in N(v)} (w'_{vi} + w'_{vj}) = |N(v)|w'_{vi} + \sum_{v_j \in N(v)} w'_{vj} \tag{4.17}$$

$$\sum_{v_j \in N(v)} l(v_i, v_j) = \sum_{v_j \in N(v)} (w_{ui} + w_{uv} + w_{vj}) \tag{4.18}$$

Identifying $l(v_i, v_j)$ and $l'(v_i, v_j)$, it is possible to write the equation for v_i which is in set $N(u)$. Based on the main principle, replacing u and v with v' should not change the mean length of the shortest path. The idea results in the lemma that implies that the summation of the shortest path lengths that contain v_i should remain the same after merging. From equations 14 to 18 we have

$$\begin{aligned}
&\sum_{v_j \neq v_i, v_j \in N(u)} l'(v_i, v_j) + \sum_{v_j \in N(v)} l'(v_i, v_j) + \sum_{v_j \in N(uv)} l'(v_i, v_j) \\
=&(|N(u)| - 1)w'_{vx} + |N(uv)|w'_{vx} + |N(v)|w'_{vx} \\
&+ \sum_{v_j \neq v_i, v_j \in N(u)} w'_{vj} + \sum_{v_j \in N(uv)} w'_{vj} + \sum_{v_j \in N(v)} w'_{vj} \\
=&(k-2)w'_{vi} + \sum_{v_j \in N} w'_{vj}
\end{aligned} \tag{4.19}$$

This equation presents the left side of Eq. 4.12 that contains to be determined variables such as w'_{vi}. On the other hand, the right side of Eq. 4.12 is constant and can be calculated based on the edge weights in the original graph. To simplify the equations, let C_i be the constant value in the equation of node v_i:

$$C_i = \sum_{v_j \neq v_i, v_j \in N(u)} l(v_i, v_j) + \sum_{v_j \in N(uv)} l(v_i, v_j) + \sum_{v_j \in N(v)} l(v_i, v_j) \tag{4.20}$$

Based on equations 12, 19, and 20, we have:

$$(k-2)w'_{vi} + \sum_{v_j \in N(u)} w'_{vj} + \sum_{v_j \in N} w'_{vj} = C_i \tag{4.21}$$

To sum up, for each node such as v_i, if the new weights are defined in a way that satisfy Eq. 4.21, the mean value of the shortest path lengths crossing through v_i does not change in the coarse graph.

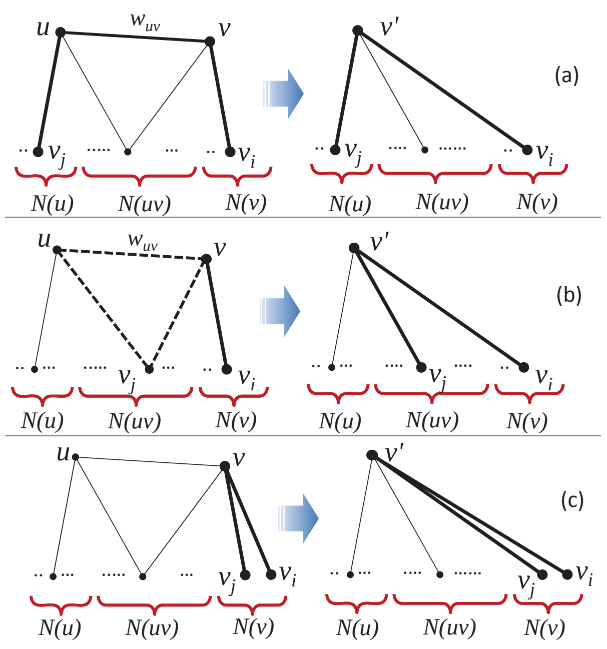


Figure 4.6: When $v_i \in N(v)$, the shortest paths that crosses v_i and v_j can be categorized into three cases: (a) $v_j \in N(u)$, (b) $v_j \in N(uv)$, and (c) $v_j \in N(v)$. (See rows 4-6 in Table 4.3)

Equations for $v_i \in N(v)$

Let v_i belong to set $N(v)$ that is connected to v not u. Similar to the previous subsection, we want to define right value for w'_{vi} and keep the average shortest path lengths that cross through v_i. Based on Figure 4.6, $l'(v_i, v_j)$ and $l(v_i, v_j)$ are computed and reported in Table 4.3. Based on lemma and Eq. 4.12, we have the following equation for node $v_i \in N(v)$.

$$(k-2)w'_{vi} + \sum_{v_j \in N} w'_{vj} = C_i \tag{4.22}$$

where C_i is constant and equal to the sum of $l(v_i, v_j)$ when $v_j \in N(v)$.

Equations for $i \in N(uv)$

The formulas for $v_i \in N(uv)$ are shown in the last three rows in Table 4.3 (Figure 4.7). Based on the main principle and the lemma, we have:

$$(k-2)w'_{vi} + \sum_{v_j \in N} w'_{vj} = C_i \tag{4.23}$$

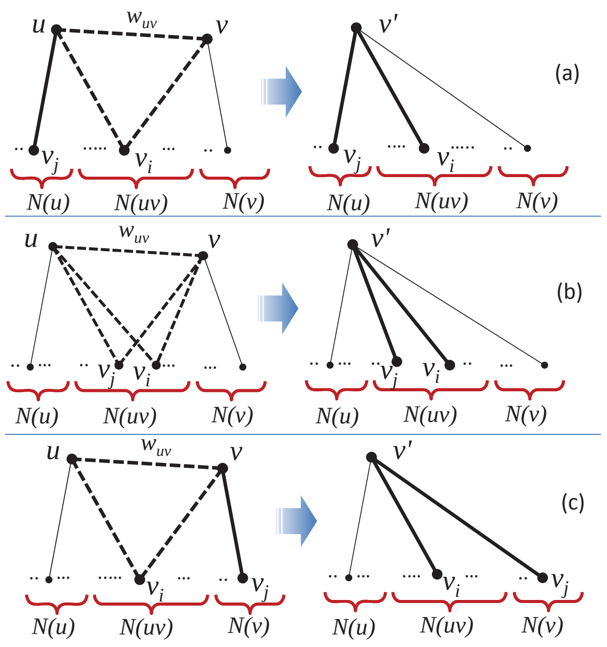


Figure 4.7: When $v_i \in N(uv)$, the shortest paths that crosses v_i and v_j can be categorized into three cases: (a) $v_j \in N(u)$, (b) $v_j \in N(uv)$, and (c) $v_j \in N(v)$. (See rows 7-9 in Table 4.3)

Similarly, C_i is the constant and it is equal to the sum of $l(v_i, v_j)$ when $v_j \in N(uv)$.

Equations 21, 22, and 23 are similar. However, for each equation, the constant part of the equations should be computed separately.

4.6.2 Solving the equations

In the previous section, a system of k linear equations is defined to find the new weights. Specifically, for merging a couple of nodes, a system of linear equations should be solved which is a costly process. The complexity of solving a system of k linear equations scales with k^3, making the approach impractical for very large graphs. Fortunately, the proposed system of equations is not a general one and in this section, we demonstrate a straightforward way to solve the equations with the complexity of $O(k)$. Hence, there is no need to solve a general linear system of equations for each merging which is a time-consuming task.

To solve the equations, we define S as the sum of the weights of the new edges.

$$S = \sum_{v_j \in N} w'_{vj} \tag{4.24}$$

Table 4.3: $l(v_i, v_j)$ and $l'(v_i, v_j)$ in different cases. All the sums are over v_j. (v_i is fixed)

	v_i	v_j	$\sum_{v_j \neq v_i} l(v_i, v_j)$	$\sum_{v_j \neq v_i} l'(v_i, v_j)$
1	$N(u)$	$N(u)$	$\sum_{v_j \neq v_i}(w_{ui} + w_{uj})$	$\|N(u)\|w'_{vi} - w'_{vi} + \sum_{v_j \neq v_i} w'_{vj}$
2	$N(u)$	$N(uv)$	$\sum(w_{ui} + min(w_{uj}, w_{uv} + w_{vj}))$	$\|N(uv)\|w'_{vi} + \sum w'_{vj}$
3	$N(u)$	$N(v)$	$\sum(w_{ui} + w_{uv} + w_{vj})$	$\|N(v)\|w'_{vi} + \sum w'_{vj}$
4	$N(v)$	$N(u)$	$\sum(w_{uj} + w_{uv} + w_{vi})$	$\|N(u)\|w'_{vi} + \sum w'_{vj}$
5	$N(v)$	$N(uv)$	$\sum(w_{vi} + min(w_{vj}, w_{uv} + w_{uj}))$	$\|N(uv)\|w'_{vi} + \sum w'_{vj}$
6	$N(v)$	$N(v)$	$\sum_{v_j \neq v_i}(w_{vi} + w_{vj})$	$\|N(v)\|w'_{vi} - w'_{vi} + \sum_{v_j \neq v_i} w'_{vj}$
7	$N(uv)$	$N(u)$	$\sum(w_{uj} + min(w_{ui}, w_{uv} + w_{vi}))$	$\|N(u)\|w'_{vi} + \sum w'_{vj}$
8	$N(uv)$	$N(uv)$	$\sum_{v_j \neq v_i} min(w_{vj} + w_{vi}, w_{uj} + w_{ui}, w_{vj} + w_{uv} + w_{ui}, w_{vi} + w_{uv} + w_{uj})$	$\|N(uv)\|w'_{vi} - w'_{vi} + \sum_{v_j \neq v_i} w'_{vj}$
9	$N(uv)$	$N(v)$	$\sum(w_{vj} + min(w_{vi}, w_{uv} + w_{ui}))$	$\|N(v)\|w'_{vi} + \sum w'_{vj}$

By replacing the sum of the new weights with S in Equations 21, 22, and 23, we have the system of k linear equations.

$$\begin{cases} (k-2)w'_{v1} + S = C_1 \\ \vdots \\ (k-2)w'_{vi} + S = C_i \\ \vdots \\ (k-2)w'_{vk} + S = C_k \end{cases} \tag{4.25}$$

S can be computed by summing up all the k equations.

$$(k-2)\Big(\sum_{v_i \in N(u)} w'_{vi} + \sum_{v_i \in N(uv)} w'_{vi} + \sum_{v_i \in N(v)} w'_{vi}\Big) + kS = C \tag{4.26}$$

$$(k-2)S + kS = C \tag{4.27}$$

$$S = \frac{C}{2k-2} \tag{4.28}$$

where C denotes the sum of the constant values in Eq. 4.25. After identifying S, the new weights can be calculated. Eq. 4.29 provides new weights if $k > 2$.

$$w'_{vi} = \frac{C_i - S}{k - 2} \tag{4.29}$$

Thus, the system of linear equations is solved in $O(k)$. Defining the equations requires the complexity of $O(k^2)$ because each of the constant values in Eq. 4.25 requires $O(k)$ to be determined. Therefore, the complexity of our method to merge one pair of nodes is $O(\sigma^2)$. The number of mergings is $(1-CR)|V|$ which makes the total complexity equal to $O(\sigma^2|V|)$. In large graphs, it is common that $\sigma << |V|$. In this case, the total complexity is $O(|V|)$.

Special cases($k \leq 2$)

If $k = 2$, then both of the equations become the same. In this case, the system of equations has infinite solutions because there are two variables and only one equation. Specifically, every set of values for w'_{vi} and w'_{vj} that satisfy the following equations lead to a zero-mean error.

$$w'_{vi} + w'_{vj} = S = C_i \tag{4.30}$$

$$w'_{vi} = w'_{vj} = C_i/2 \tag{4.31}$$

$\tilde{q}_i$, or $\tilde{r}_i$. Figure 4.8(a) illustrates the situation when $|N(u)| = |N(v)| = 1$.

If $k = 1$, then u and v cannot be part of a shortest-path unless they are the source or destination nodes. Here, for finding the new weight, we apply the same fact used for *Shrink*: for the least mean square error, the mean of error has to be zero. In fact, the w'_{vi} should be equal to the mean of the paths passing through v_i. Similarly, the probabilities of starting the paths from v and u are assumed to be the same (i.e 1/2). Figure 4.8(b) illustrates the situation when $|N(u)| = 1$. The following equations are applied to assign the new weights for $k = 1$.

$$|N(u)| = 1,\ w'_{vi} = \frac{w_{ui}}{2} + \frac{w_{ui} + w_{uv}}{2} = w_{ui} + \frac{w_{uv}}{2} \tag{4.32}$$

$$|N(uv)| = 1,\ w'_{vi} = \frac{w_{ui}}{2} + \frac{w_{vi}}{2} = \frac{w_{ui} + w_{vi}}{2} \tag{4.33}$$

$$|N(v)| = 1,\ w'_{vi} = \frac{w_{vi}}{2} + \frac{w_{uv} + w_{vi}}{2} = w_{vi} + \frac{w_{uv}}{2} \tag{4.34}$$

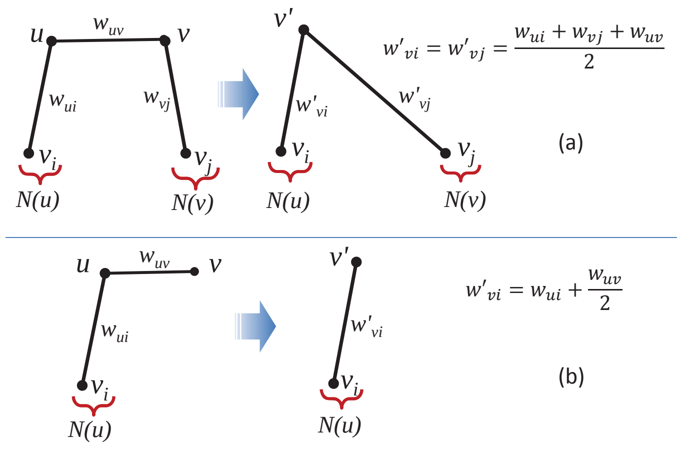

u
w_{uv}
v
w_{ui}
w_{vj}
v_i
v_j
$N(u)$
$N(v)$
v'
w'_{vi}
w'_{vj}
$w'_{vi} = w'_{vj} = \frac{w_{ui} + w_{vj} + w_{uv}}{2}$
(a)
$w'_{vi} = w_{ui} + \frac{w_{uv}}{2}$
(b)

Figure 4.8: $k = 2$, $|N(u)| = 1$, $|N(v)| = 1$. The sum of new weights should be $p_1+q_1+w_{uv}$. (b) $k = 1$, $|N(u)| = 1$. The figure shows u, v, and v' when $|N(u)| = 1$.

To sum up, if $k = 2$ or $k = 1$, equations 31-34 determine the new weights. If $k = 0$ then we do not need to define new weights and v' will be a single node.

Algorithm 4.1: Node selection algorithm

```
  Input: $v$, $u$, $\Theta$
1 −$v$, $u$: candidate pair for merging
2 −$\Theta$: threshold
  Output: Acceptance/Rejection of the pair, updated $\Theta$
3 if $(N(u) + N(uv)) \times (N(v) + N(uv)) < \Theta$ then
4     $\Theta = \Theta - 1$;
5     return true;
      /* the pair is accepted */
6 else
7     $\Theta = \Theta + 1$;
8     return false;
      /* the pair is rejected */
9 end
```

4.6.3 Node selection criteria

To minimize the error, we define the following rule for the nodes that are going to be merged (i.e. v and u): *the node pair is desirable for merging if it has a small number of neighbours.* Specifically, given two nodes u and v, we define the multiplication of the number of the neighbours of the nodes, as a way for node selection. Hence, if the result is small enough, the selected nodes are suitable for merging. The simple way is setting a constant threshold, Θ, on the multiplication result that has two drawbacks. First, it makes the method parametric. This means the threshold should be defined beforehand and the appropriate threshold is different for each graph. Second, by using threshold, the algorithm may fall into an infinite loop. After merging v and u, the average degree increases because the degree of v' is often bigger than the degree of v and u. Therefore, after plenty of merging, there may be no pair of nodes that satisfies the threshold condition.

To solve this problem, we update the threshold after each selection. The update is based on whether the current pair satisfies the threshold condition or not. Specifically, after picking a pair of nodes, if $N(u) \times N(v) > \Theta$ then theta increases by one otherwise it decreases by one. Based on this criteria, the candidate pairs are mainly from those with the smaller degree. Algorithm 1 shows the proposed method for node selection. The algorithm starts with $\Theta = 1$.

The proposed node selection criteria improve the accuracy of *Shrink*. The improvement results from the probability distribution of occurring the neighbors of v and u in shortest paths. In our model, the uniform probability distribution function is used which is closer to reality when merging nodes with few neighbors.

4.7 Executing and refining stages

The output of the coarsening stage is the coarse graph in which distances between nodes are almost the same as those in the original graph. As each node in the coarse graph is a supernode containing some nodes, there is a mapping between the nodes of the coarse and original graphs. Therefore, instead of querying a pair of nodes in the original graph, we can query the equivalent pair in the coarse graph.

Distance-based algorithms run faster on the coarse graph because it is smaller. The smaller the coarse graph is, the faster a query executes. However, the accuracy drops by decreasing the size of the coarse graph. As one node is eliminated in each merging, the number of nodes in the coarse graph is equal to the number of nodes in the original graph minus the number of mergings. The reduction in the number of edges in each merging depends on the number of the common neighbours of the merged nodes. Specifically, the number of edges is reduced by $|N(uv)| + 1$ in each merging.

Integrating with any distance-based algorithms, *Shrink* can improve the performance. To investigate the effect of *Shrink* on the performance, the complexity of the distance-based

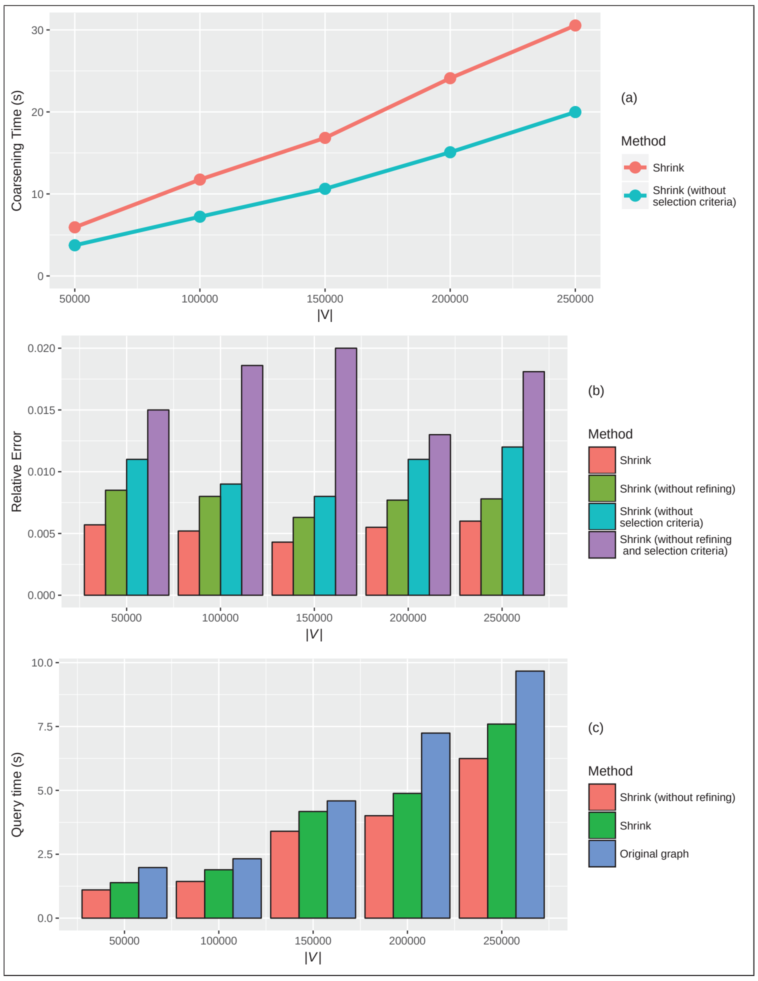


Figure 4.9: Results for the subgraphs of NY road network with $CR = 50\%$ (a) coarsening time, t_c (b) average relative error for different setups (c) Shrink *query time with and without refining stage, t_{q2}, and the query time on the original graph t_{q1}.*

algorithms should be taken into account that depends on the number of nodes and edges. For example, if the complexity of an algorithm is $O(|V|^2)$ and $CR = 50\%$, it runs four times faster on the coarse graph.

The output of the executing stage is a path in the coarse graph that consists of the source and destination nodes. However, the actual shortest path is still unknown because a supernode on the path is representative of several nodes. In the refining stage, the path is projected from the coarse graph to the original graph. As a result, a small subgraph of the original graph is obtained. Each node in this subgraph belongs to one of the supernodes of the extracted path. Rerunning the algorithm on the extracted subgraph specifies the laying nodes on the shortest path in the original graph. Furthermore, it provides a better estimation of the distance. It should be noted that rerunning the algorithm on the subgraph is much faster than running the algorithm on the original or coarse graph because the subgraph is very smaller than the whole graph.

The refining stage is an optional stage. In some applications, just a fast estimation of the length of the shortest path is needed. Hence, the refining stage is not necessary. Furthermore, sometimes there is not enough space to store the original graph. In these cases, we can discard the original graph and work on the coarse graph instead. For example, maintaining APSP results on disk for quick lookups requires $O(|V|^2)$ space which is unfeasible in many applications. In this case, *Shrink* reduces the storage size by CR^2 by processing the coarse graph.

4.8 Experiments

In this section, we evaluate *Shrink* in terms of time complexity and accuracy. Furthermore, we investigate the impact of the graph size, compression ratio, graph type, and node selection criteria. We compare *Shrink* performance with one of the state-of-the-art graph summarisation method introduced in [Toivonen et al., 2011]. We choose this method as a baseline because it is compatible with weighted graphs. Furthermore, it aims to preserve the query over all node pairs and hence it is comparable with our method.

4.8.1 Experiment setting

Most of the experiments run on New York City road network (NY road network), an undirected weighted graph with 264,346 nodes and 733,846 edges[2]. In this graph, a node represents an intersection or a road endpoint and the weight of an edge represents the length of the corresponding road segment. Furthermore, to investigate the performance of our approach on different datasets, we examine graphs from Stanford network data collections including friendship network, collaboration network, web graph and social network in section 4.8.6 [Leskovec]. The experiment runs on PC with the configuration of Intel(R) Core i7, 3.4GHZ and 8G RAM. The PC runs Windows 7 and JDK 7 that runs on NetBeans 7.4. The algorithm has been implemented using the free Java graph library JGraphT, which includes mathematical graph-theory objects and algorithms [Naveh].

[2]http://www.dis.uniroma1.it/ challenge9/download.shtml

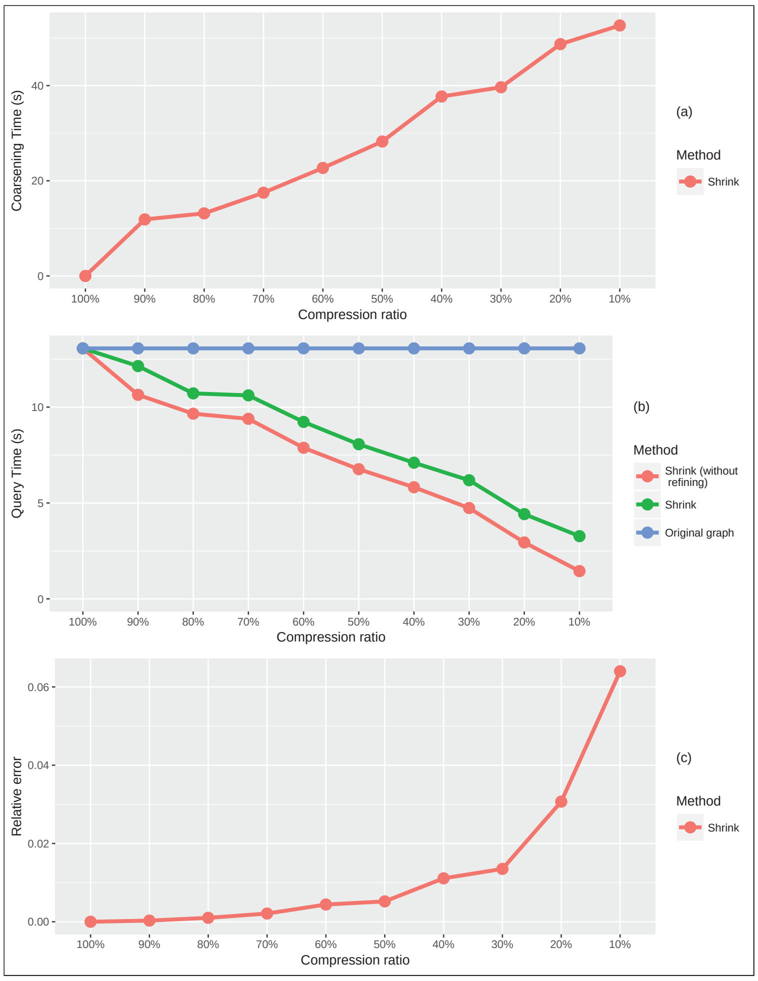


Figure 4.10: Results for the subgraphs of NY road network. (a) coarsening time, t_c (b) Query times t_{q2} and t_{q1} (c) Average relative error

The actual distances in the original graph are compared to the outputs of *Shrink*. The following parameters are measured in the evaluation procedure: 1) coarsening time (t_c): the required time to produce coarse graph 2) original graph query time (t_{q1}): the query time for 100 random pairs before compression 3) *Shrink* query time (t_{q2}): the query time for the same 100 pairs using *Shrink* (4) relative error: the average relative error over the 100 pairs.

4.8.2 Primary results

In this set of experiments, the effect of our compression method on NY road network is investigated while the compression ratio is 50%. Specifically, we apply Dijkstra algorithm on

NY road network and its coarse graph and evaluate the impact on the distances. We run this experiment 5 times, and each time only a part of the graph is loaded. For covering graphs with different sizes, the number of loaded nodes ranges from 50,000 to 250,000 by a step of 50,000.

Figure 4.9(a) shows the running time for the coarsening stage, t_c. It takes only 20 seconds to compress the whole graph to half. When applying node selection criteria, t_c increases to 31 seconds because some selected pairs are not merged and are discarded. The trend is linear because coarsening time has a linear relationship with the number of merged nodes. Figure 4.9(b) shows the relative error with and without the refining stage and node selection criteria. Node selection criteria have a bigger effect on the relative error compared to applying refining stage. However, considering node selection criteria, we need more time to compress the graph. Figure 4.9(c) shows the improvement in the query time caused by *Shrink* with and without the refining stage.

4.8.3 Impact of compression ratio

The compression ratio affects the speed, required storage, and accuracy of *Shrink*. When the compression ratio increases, the new graph has fewer nodes, and hence the shortest-path algorithms run faster but with lower accuracy. Therefore, the performance is adjustable which means for a given graph, the compression ratio should be set based on the desired speed and storage and acceptable error rate. Specifically, we can continue merging until the desired performance is achieved.

In this experiment, the original graph is NY road dataset (with 264,346 nodes) and the compression ratio changes from 10% to 100%. Results are reported in Figure 4.10. The coarsening stage running time for different compression ratios is shown in Figure 4.10(a). The trend is not quite linear because the average degree has an increase after compressing the graph. As discussed, the average degree in the graph affects coarsening time. Figure 4.10(b) shows how the query time changes when compression ratio increases. Without using *Shrink*, the query time for the original graph lasts for 13 seconds. When the number of nodes in the coarse graph decreases, the queries run faster. Figure 4.10(c) demonstrates the effect of the compression ratio on the average relative error for *Shrink*. It can be inferred that the more the nodes are merged, the less accurate the estimated lengths are. Figure 4.10 validates the effectiveness of the proposed approach. As an illustration, if the size of the graph three to 20%, SSSP algorithms runs three times faster while the average error is only 3%. In this

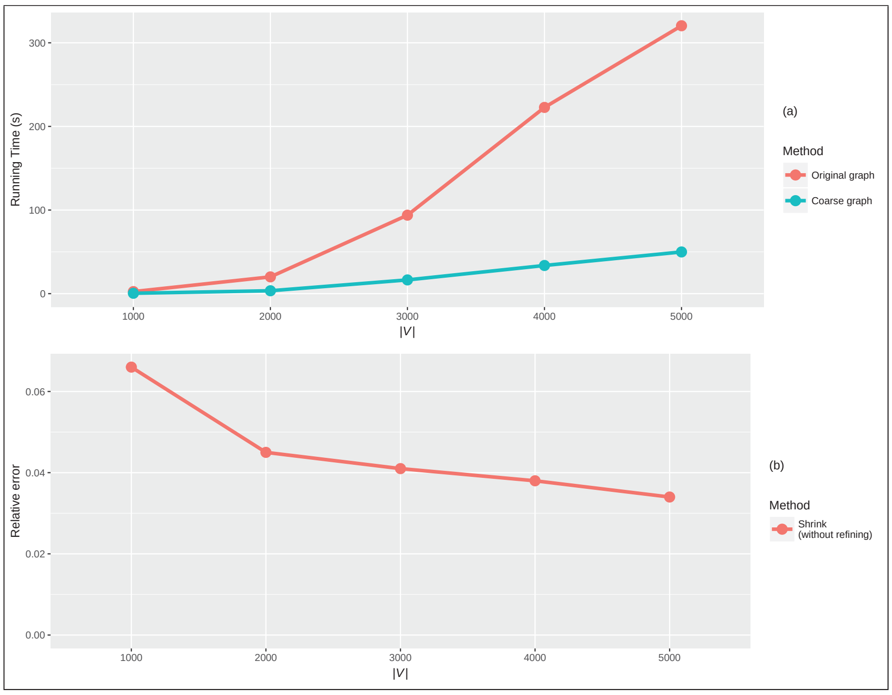


Figure 4.11: Computing APSP for the subgraphs of NY road network with CR = 50%. (a) required time for the original and coarse graphs (b) the average relative error over all pairs

case, by using the refining stage, the nodes on the shortest path are also provided.

4.8.4 Impact of graph size

The evaluation in this subsection is different from that in other subsections. Here, we assess the performance of *Shrink* using All-Pair Shortest Path (APSP) method while in other experiments we use single-source shortest-path (SSSP) method and compare the distances between 100 pair of nodes in the original and coarse graphs. For APSP, the average error over all distances is calculated, but it cannot be applied to a graph with millions of nodes. The Floyd Warshall, a classic APSP algorithm, computes all shortest paths between each pair of nodes with $O(n^3)$ comparisons. Considering the heap size and computational cost, it is not possible to apply Floyd Warshall algorithm on the whole NY dataset. Thus, in this experiment, the number of nodes ranges from 1000 to 5000 by a step of 1000. Figure 4.11(a) has two trends. The first one shows the Floyd Warshall running time for the original graph and the second one shows *Shrink* total running time including the coarsening, executing and refining stages. Figure 4.11(b) shows the average relative error over all shortest paths. It can be seen that for improving the speed of the algorithm up to 6 times, we only lose less than 3 percent of the accuracy. For the larger graph, the estimated distance is more accurate because the paths are long and according to section 4.5.2, the effect of our compression method on the long paths is little.

The required storage and heap size is another challenge while working with the large graph. As the coarse graph has less node and edges, it can be stored in less space. In our experiments, the maximum heap size of the virtual machine is 512MB which can handle only 12500 nodes to run Floyd-Warshall algorithm. However, if the graph is reduced by half, it can handle up to 24000 nodes.

4.8.5 Comparison with a state-of-the-art method

Here we compare *Shrink* with Compression of Weighted Graph (CWG) method introduce by Toivonen et al. [Toivonen et al., 2011]. CWG is based on merging nodes and edges to supernodes and superedges. Toivonen et al. define the difference between the edge wights and the superedges weights as the distance between the original graph and the coarse graph. They claim that the distance is minimized when the superedge weight is the mean of the original edge weights. Therefore, the main focus is on the common neighbours of u and v (i.e $N(vu)$). They also consider the number of the nodes that u and v include. Based on our notations, Equation 4.35 gives the new weights for CWG.

$$w'_{vi} = \frac{|u|w_{ui} + |v|w_{vi}}{|u| + |v|}, \; vi \in N(uv) \tag{4.35}$$

where $|v|$ is the number of original nodes in supernode v.

To make the comparison fair, we run *Shrink* without refining stage. We also do not consider node selection criteria and the same set of node pairs are merged in both algorithms. Figure 4.12(a) shows that it takes approximately the same time to compress the graph with either of methods. The reason is that in each merging both algorithms focus on one pair of nodes and spend the same time to find new edge weights. However, it can be seen from figure 4.12(b) and 4.12(c) that our method outperforms CWG. While the average relative error is around 2% for our method with $CR = 50\%$, this value is around 18% for CWG. The reason is that *Shrink* takes more parameters into account compared to CWG. For assigning a new weight to the edge between two nodes, *Shrink* considers not only the edge weights between the nodes in the original graph but also the edge weights of their neighbours.

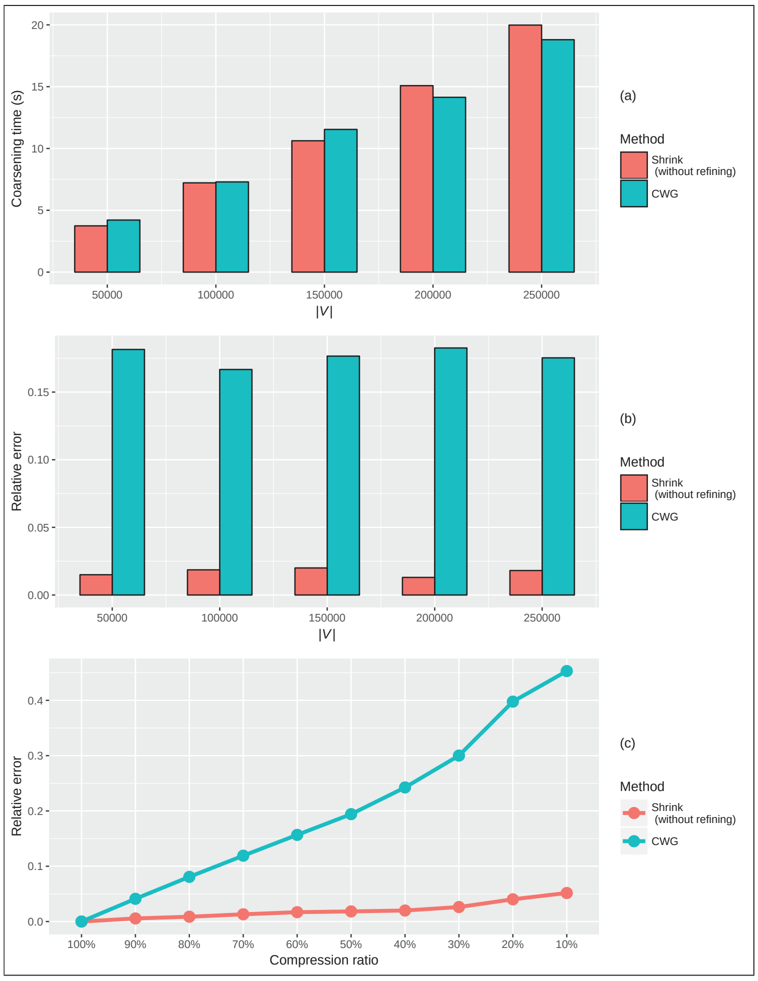


Figure 4.12: Comparison between Shrink and CWG (a) t_c for NY road network subgraphs with different size ($CR = 50\%$) (b) average relative error for NY road network subgraphs with different size ($CR = 50\%$) (c) average relative error

Table 4.4: Experiment Results for different datasets with $CR = 20\%$

Input Graph				Results			
Name (type)*	$\lvert V\rvert$	$\lvert E\rvert$	Diameter	t_c	t_{q1}	t_{q2}	RE
DBLP (collaboration)	317,080	1,049,866	21	62s	31s	10s	19%
Brightkite (friendship)	58,228	214,078	14	12s	7s	3s	9%
Epinions (social)	75,879	508,837	14	27s	13s	9s	9%
New York (road)	264,346	365,050	1589	49s	13s	6s	3%
Notre Dame (web)	325,729	1,497,134	46	97s	20s	8s	2%
Florida (road)	1,070,376	1,343,951	2957	181s	52s	17s	0.5%
Great Lakes (road)	2,758,119	3,397,404	4145	629s	151s	54s	0.4%

* t_c: Coarsening time, t_{q1}: Query time for the original graph, t_{q2}: *Shrink* query time, RE:Relative Error

Table 4.5: summarisation results for four users when the compression ratio is 10.

	#days	Original Graph nodes/edges	Coarse Graph supernodes/edges	Error
U1	466	117/298	12/22	4%
U2	668	161/373	17/35	3%
U3	454	81/140	9/16	6%
U4	81	47/98	5/12	7%

4.8.6 Impact of graph type and graph density

In addition to road networks, we apply our method to different types of graphs including friendship network, collaboration network, web graph and social network. These datasets can be downloaded from Stanford network data collections [Leskovec]. Since the graphs are unweighted, all of the edges are assigned the same weight and the direction of the edges are not considered. We also apply *Shrink* to larger road network including Florida and Great Lakes road network with 1 and 2.7 million nodes, respectively. In this experiment, the reduction ratio is set to 20%.

Table 4.4 shows the results. As expected, when the input graph is dense, the accuracy of our method drops. For example, Brightkite friendship network is a very dense graph because the diameter (longest shortest path) is just 14. As in our model, we assume that the input graph is large and has long paths, *Shrink* provides less accurate compression compared to NY road network, which is a quasi-planner sparse graph. Furthermore, in Brightkite network, more than 40% of the nodes have only one neighbour that drops the accuracy. The reason is that our model tries to minimize the effect on the paths that pass through the supernodes while no path passes through a one-degree node (unless the node is the start or end node). *Shrink*s compresses larger graphs better because they usually have longer shortest paths. According to Table 4.4, *Shrink* is practical for large graphs as it can compresses a graph with 2.7 million nodes into fifth in 10 minutes.

4.8.7 Summarisation effect on movement graphs

Here, we investigate the effect of *Shrink* on two human mobility mining algorithms in terms of time, accuracy, and granularity. After constructing the movement graph, we apply *Shrink* and consequently, we obtain the coarse graph. we apply the mobility mining algorithms before and after summarisation to investigate the summarisation effect.

Movement graph construction

In this section, we describe the procedure of constructing the movement graph from a real-world dataset, Device Analyser [Wagner et al., 2014]. The Device Analyser dataset includes data gathered from running background processes, wireless connectivity, GSM communication, and some system status and parameters. In this dataset, MAC addresses, Wifi SSIDs, and other forms of identification are hashed due to privacy purposes. As a result, there is no ground truth or information about the geography and semantic of the locations [Wagner et al., 2014]. In our experiments, we extract the cell tower IDs (CIDs) connected to the smartphone. A CID is a labeled location whose geographical coordinates is unknown.

To construct the movement graph, we start with identifying the stay points that can be defined in different ways. Based on our definition, a CID is considered as a stay point if the user is connected to it longer than one hour. A node is assigned to each stay point, and two nodes are connected if the user moves from one to the other stay point. The weight of the edge between the nodes denotes the traveling time from the time the user leaves the source stay point until he arrives at the destination one. If the user commutes between two nodes more than once, the mean values of traveling times is considered as the weight of the edge.

A movement graph belongs to a user movement during a specific period. Table 4.5 shows the details of movement graphs extracted from Device Analyser. The size of the graphs and the amount of the processed data are reported.

Movement graph summarisation

After building the user's movement graph, we apply Shrink to summarise the graph. Shrink, applicable to a weighted graph, reduces the size of the graph by merging nodes while trying to preserve the distance between the nodes. After merging two nodes, a set of new weights are assigned to the edges connected to the nodes in the way that the distances have the least change. Hence, the distances between the nodes in the original and coarse graphs are almost the same. Therefore, to find the traveling distance between two CIDs, it is possible to run the shortest path algorithm on the coarse graph instead of the original graph. Each node in the coarse graph is a supernode representing a cluster of nodes from the original graph.

Shrink is flexible about compression ratio (CR), which is the ratio of the number of nodes in the original graph to the number of nodes in the coarse graph. However, when the compression ratio is high, the distances are not preserved well. As the compression ratio is flexible, the user can set it based on the available storage. After storing the coarse graph, it can be queried without decompression. For any distance-based query such as shortest path query, the result is almost the same as running the query on the original graph.

In our application, a supernode in the coarse graph represents a cluster of CIDs that may belong to a specific location such as university or shopping center. For example, assume several CIDs exist in the university and the user does not always connect to the same one. Shrink merges all the CIDs in the university to a supernode and updates its new edge weights.

Table 4.5 reports the error when the compression ratio is equal to 10. The error is the average relative difference between the distances in the original and coarse graphs over 100 node pairs. The small value of the error indicates that the distances have been preserved well. We run our experiments on 4 different users containing 1689 days of data in total.

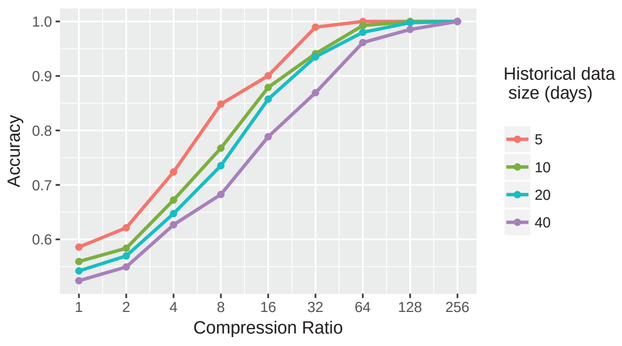
Effect on prediction algorithms

Here, the goal is to predict the location of the user based on the historical data and the time of the day. Specifically, we compute the probabilities of the presence of the user in all locations at a specific time of the day (e.g. 6 pm) by processing the historical data. Then, we choose the location with the highest probability as the location of the user. This method is one of the baselines for location prediction [Do et al., 2015]. In our experiment, the Device Analyser dataset is used where the locations are labeled with CIDs.

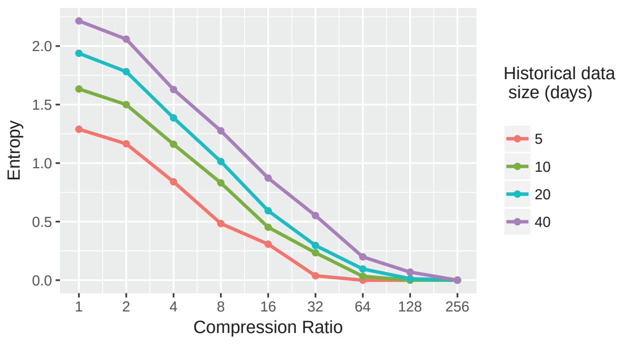
After summarisation, the prediction algorithm is applied to the coarse graph. The number of experimental instances is 1520 from 4 Device Analyser users mentioned in Table 4.5. For each instance, the location of the user is predicted at a specific time that is identified by a supernode. If the predicted supernode in the coarse graph includes the actual CID that the user is connected to, the prediction is considered as successful. The accuracy is the number of successful predictions divided by the total number of experimental instances.

The compression ratio (CR) ranges from 1 to 256. CR equal to 1 indicates the case that no summarisation is performed and the experiments run on the original graph. On the other hand, when CR is equal to 256, all of the nodes are merged into a single node and all information is lost. The length of the historical data ranges from 5 to 40 days.

The accuracy is reported in Figure 4.13 (a). As it can be seen, when the compression ratio is high, the accuracy increases because the number of the possible locations for the user decreases which makes the prediction easier. In Figure 4.13 (b), Shannon entropy is reported. Intuitively, the entropy denotes the average amount of information gained by knowing the user's location. From the figure, it can be seen that when the compression ratio increases, the



(a)



(b)

Figure 4.13: Effect of the summarisation on the (a) accuracy of prediction (b) information gained by knowing the location of the user.

entropy decrease as there are less possible locations. When CR is 256, we gain no information by knowing the user's location as there is only one location (i.e. supernode). Furthermore, when the historical data is big, the number of CIDs visited by the user increases, the graph is larger, and the entropy is higher. Reporting the entropy is a proper way to measure how much privacy is lost by disclosing the user's location.

Effect on similarity mining algorithms

In this section, we show how the summarisation changes the performance of the methods that measure the similarity between the movement graphs. Specifically, given two movement graphs of a user related to two different periods, the aim is to compute the similarity between the graphs. The result states how much the user's mobility pattern changes over time, and it is based on the locations that the user visits and their frequencies. For example, let us consider a user that goes to the university every day but during the exam period, he goes to

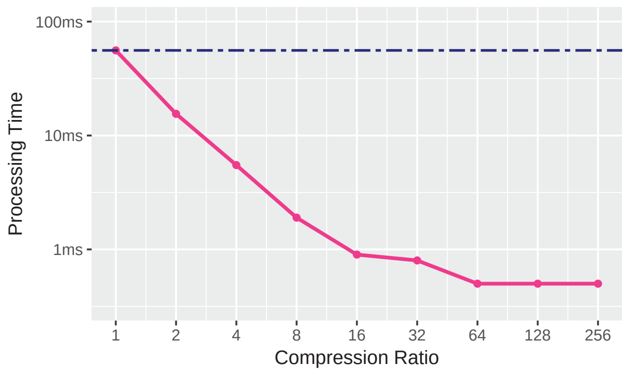


Figure 4.14: Impact of the coarse graph size on the processing time required for measuring similarity. The dashed line shows when there is no summarisation.

the library to study. By building the movement graph in exam period and comparing it with user's normal movement graph, we can detect when the user's mobility pattern changes [Sadri, 2016].

In our experiments, we deploy Graph Edit Distance (GED) to measure the similarity between two graphs [Sanfeliu and Fu, 1983]. Using this approach, the dissimilarity between two graphs is defined as the minimum number of operations required to convert one graph to the other. The operations include edge insertion/removal, node insertion/removal, and increasing/ decreasing an edge weight.

Figure 4.14 shows the effect of the summarisation on the processing time for the GED algorithm. When the compression ratio is high, the coarse graph has fewer nodes. Consequently, comparing the graphs is performed faster. In Figure 4.15, we show that comparing the coarse graphs is almost the same as comparing the original graphs. In fact, the summarisation has a little effect on the GED results and if two graphs are similar/dissimilar, after the summarisation, the similarity/dissimilarity remains the same. However, by increasing the compression ratio, the uncertainty is increased too. For CR=2, the Pearson correlation coefficient is 0.98 but when the compression ratio becomes bigger, the correlation decreases. To sum up, by increasing the compression ratio, the processing time of GED algorithm decreases but at the same time, the uncertainty of the result increases, which means the results is less reliable.

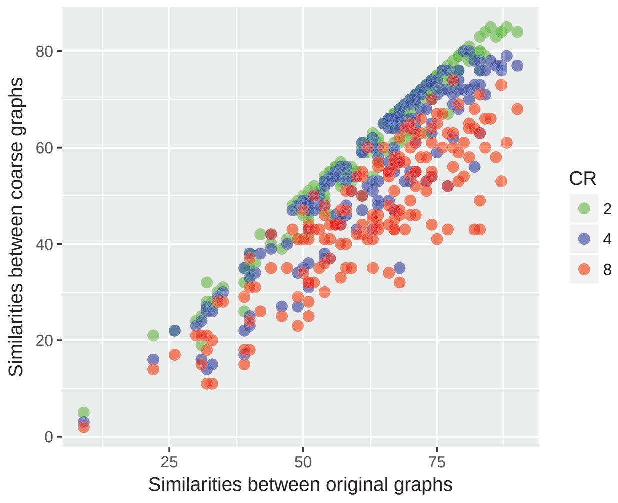


Figure 4.15: By increasing the compression ratio (CR), the correlation decreases.

4.9 Conclusion

In this chapter, we introduce *Shrink* as a new distance preserving graph summarisation method. In the first stage, the graph size is reduced by replacing node pairs with supernodes. The queries run more efficiently on the reduced graph, called coarse graph. To find the exact shortest path including the laying nodes on the path, we rerun the query on a subgraph of the original graph. *Shrink* is a generic compression method that can be applied to different types of the graph. In the experiment section, we investigate the impact of different parameters on the performance. We also deploy *Shrink* to summarise the movement graph to produce an abstract (coarse) graph easy to be processed and queried.

Although *Shrink* is quite generic, our main focus in this chapter are the graphs that contain spatial information of the user's mobility such as road network and connectivity data. We evaluate our graph summarisation algorithm by applying to New York road network in which the nodes represent the intersections and edges represent distances. This graph have the user's mobility information when he moves from one intersection to another. Furthermore, we also apply *Shrink* to connectivity data from Device Analyser dataset in which the nodes represent the CIDs and edges shows the changes in the connection. After summarisation, each supernode in the road network represents several intersections while in the connectivity data each node represents several CIDs.

The performance of *Shrink* depends on how much the differences between the distances in the coarse and original graphs. In the experiments, we investigate the impact of graph type, graph size, and compression ratio on the performance. *Shrink* has better performance on sparse graphs with large diameters. Furthermore, we prove that *Shrink* is more suitable for larger graphs. The compression ratio has a reverse effect on the performance which means when the large compression ratio is high, the distances are not preserved well.

We specifically investigate the effect of summarisation on two algorithms related to human mobility mining: location prediction and similarity mining. First, a location prediction algorithm is applied to both original and coarse graphs. We show that there is a trade-off between the granularity and accuracy of the results. Specifically, the prediction based on the coarse graph results in a better accuracy but the output is coarse-grain that includes several locations. Second, we study the effect of the summarisation on computing the similarity between two movement graphs. To this end, we apply a graph similarity metric to both original and coarse graphs. As the coarse graph has fewer nodes, the summarisation speeds up the calculation of similarity. However, if the compression ratio is too big, we lose information and the similarity cannot be calculated correctly.

Chapter 5

Partial daily trajectory prediction

5.1 Introduction

Understanding human mobility has always been a substantial pursuit in academic research due to the multitude of potential applications, such as link prediction [Wang et al., 2011], urban planning [Gonzalez et al., 2008; Li et al., 2012], and resource management, such as wireless system management [Cheng et al., 2003] or smart home heating systems scheduling [Das et al., 2002], to name a few. It also benefits location-based service providers that deliver services to users based on their location, such as traffic updates, suggested routes, or location-based advertisement [Ribeiro et al., 2014; Bastani et al., 2011]. However, for these suggestions to be precise, there is a need for trajectory prediction containing both spatial and temporal information of the user's future movements.

While there are many techniques for human location prediction, they all have one or more limitations that have reduced their applicability for trajectory prediction in practice [Barzaiq and Loke, 2015; Barzaiq et al., 2015; Chen et al., 2014; Jeung et al., 2010; Scellato et al., 2011; Jeung et al., 2008]. This observation has motivated us to purpose a trajectory prediction approach. Given the historical data and the user's trajectory in the first part of the current day (e.g. trajectory in the morning), our approach completes the user's daily trajectory by predicting the trajectory for the rest of the day (e.g. prediction of the afternoon trajectory). The proposed approach is unique in offering all the following features:

- **Sequence of locations**: Most approaches in the literature focus on next location prediction, which is the prediction of the user's location at a certain time [Chen et al., 2014; Jeung et al., 2010; Scellato et al., 2011; Sadilek and Krumm, 2012; Jeung et al., 2008]. Although next location prediction approaches are designed to predict only one location, they can predict the sequence of locations by multiple implementations (e.g. every one hour) [Do et al., 2015]. In this case, the departure times are not estimated and the duration of stays is assumed to be constant. In contrast, our approach inherently returns the sequence of visited locations and can predict the location at any particular time in the future.

- **Departure times**: Some location prediction approaches focus on only the next location that the user visits regardless the time elapsed [Gambs et al., 2012; Gidófalvi and Dong, 2012]. The departure times from the locations are not estimated in these approaches. In contrast, our method can predict both the location and time of the departure from each point and all the following sequences.

- **Granularity**: The past location prediction methods largely rely on discretizing the trajectory first, and then returning regions rather than accurate locations [Ashbrook and Starner, 2003; Chen et al., 2014; Morzy, 2007; Monreale et al., 2009; Lei et al., 2011]. For example, the prediction model is trained on GPS data, but the predicted location is a cell grid rather than GPS coordinates [Chen et al., 2014]. In our approach, the predicted trajectory has the same granularity as the training/historical data.

- **Generality**: Most prediction techniques are designed for a specific type of data. Some handle labelled locations such as WiFi access points where the geographical information (e.g. latitude and longitude) may not be available while others process geographical locations such as GPS data. The proposed method in this chapter is able to handle both labelled and geographical trajectories.

- **Diversity of the predicted locations**: Some existing methods predict the location from a finite set of locations, such as significant locations (e.g. home and office) or stay points, while discarding other locations [Ashbrook and Starner, 2003]. For example, Eagle et al. consider only four discrete locations for prediction [Eagle and Pentland, 2009]. In contrast, our method does not ignore any location that exists in the historical trajectories.

Table 5.1 summarizes the differences and shows the unique aspects of trajectory prediction approaches. Prediction of the departure times in which the lengths of the stays are estimated [Lee and Hou, 2006; Meng et al., 2015] has also been tackled by previous research and is mentioned in the table. Table 5.1 also illustrates the sample outputs for a typical scenario. Assume that the user is at the office at 3 pm and wants to buy groceries on his way home from the office. What location prediction and departure time approaches can contribute to a recommendation system is to advise the departure time (i.e. 4 pm) and the location at 6 pm (i.e. home). Predicting the user's location, the recommendation system suggests only places near the home or office for shopping. However, having the trajectory of the user between the office and home, the recommendation system can also suggest the closest shops on the route home for the user. It can also check if that shop is open at the time the user will reach the shop.

Despite its importance to applications, none of the existing works has focused on the completion of a partial daily trajectory of a user while predicting spatial and temporal sequences with fine granularity for both labelled and geographical trajectories. Routing and traffic recommendations would greatly benefit from such an approach. For example, assume that the user is on her way to work in the morning. By predicting the path to the office and the departure time, a location-based service provider could notify her of transport disruptions in advance and suggest alternative routes with added precision. A short summary of our approach starts with a user's trajectory up to a certain time of the day. Our algorithm predicts the trajectory for the rest of the day. For example, we have a trajectory for the morning (e.g. up to 12:00) and the problem will be to predict the trajectory in the afternoon (e.g. from 12:00 to 23:59). We predict a trajectory which includes the sequence of future locations with timestamps that will be visited by the user for the rest of the day. If the trajectory is a geographic one, e.g. GPS locations, then the granularity of the prediction is at the GPS level rather than, like in other approaches, with a coarser granularity of regions [Sadri et al., 2017].

Methodologically, our approach investigates the similarities between the given sub-trajectory of the current day and the historical data for prediction. Specifically, the trajectories from the historical data similar to the given sub-trajectory play an important role in the prediction of the succeeding part of the new day's trajectory. For example, it may be inferred from the historical data that if a user goes to university early in the morning, she would leave the university early. Therefore, if the trajectory in the morning shows that the user goes to university early, a reasonable prediction should anticipate that the user does not stay at the university for a long time. Defining a comprehensive similarity metric between the trajectories is essential in the proposed method. To do this, we combine two similarity metrics by considering the spatio-temporal properties of the trajectory as well as the sequence

Table 5.1: Comparison between Location prediction, departure time prediction, and our problem (trajectory prediction)

	Location prediction	**Departure prediction**	**Partial human daily trajectory prediction**
Question to be answered	Where will the user be at a certain point in time?	When will the user depart from the current location?	What is the user's trajectory (sequence of locations and time points) for the rest of the day?
Location granularity of the output	Region (e.g. POI, gridded map, area covered by a cell tower)	N/A	Same as the input (e.g. latitude and longitude for GPS trajectories)
Sample output inference	The user is at home at 6 p.m.	The user leaves the office at 4 p.m.	The user leaves the office at 4 p.m., passes along George St., and arrives home at 5 p.m. The user stays home for the rest of the day.

of the locations. We evaluate our method with both *labelled trajectories* and *geographical trajectories*, and the results demonstrate the effectiveness and efficiency of the method compared to baselines. Using our method, the prediction error is reduced by 10% and 35% for labelled trajectories and geographical trajectories, respectively. This saves the user's time and enhances her experience. For example, assume that a recommendation system recommends the user to watch fireworks in the main square at 9 p.m. based on the prediction. However, the user arrives at 8 p.m. and she has to discard the recommendation or wait for one hour. On average, our approach reduces the waiting time to 40 minutes. Similarly, from the spatial point of view, the recommended location might be one kilometer away from the user's location. Using our method, the distance decreases to around 650 meters.

The main contributions of this chapter are as follows:

- We introduce a mechanism that combines similarity metrics over temporal sequences of locations to estimate similarity of user trajectories. This new metric provides an effective comparison between two trajectories.
- We complete the partial daily trajectory by predicting the trajectory for the rest of the day. To ensure the quality of the predictions, different components are deployed in our method, including temporal correlation, temporal segmentation, and outlier removal.
- We apply the method to two real-world datasets consisting of labelled trajectories and

geographical trajectories, and we show the reduction in the prediction error, which is the difference between the predicted trajectory and the actual trajectory. The considerable decrease in prediction error unlocks more precise spatio-temporal user recommendations in the context of location services.

In the next section, we discuss related work. In Section 5.3, we formally state the problem. In Section 5.4, we discuss the rationale behind our approach by leveraging on the salient characteristics of human mobility. Our approach for trajectory prediction is described in Section 5.5. The experiment results of our algorithm are reported in Section 5.6. Finally, Section 5.7 concludes the chapter.

5.2 Related work

Extensive research has been undertaken on mobility prediction, and it is a key issue in a large number of applications, such as mobile wireless systems [Cheng et al., 2003], road networks [Kim et al., 2007; Jeung et al., 2010], and smart homes [Das et al., 2002]. In the networking community, some researchers focus on prediction in WiFi networks, while others predict the connectivity to GSM mobile phone towers (i.e. CID) [Nicholson and Noble, 2008; Song et al., 2006]. These methods anticipate client connectivity to the network to enhance mobility management [Liu and Maguire Jr, 1996], cell assignment [Das and Sen, 1999], paging [Bhattacharya and Das, 2002], and call admission control [Yu and Leung, 2002]. In addition to the mobility pattern of smartphone users, trajectory prediction is used in road networks where movements are constrained to roads on the map [Kim et al., 2007; Jeung et al., 2010; Krumm and Horvitz, 2006]. Location prediction systems are also used in smart home environments to maximize occupant comfort and minimize operation costs [Das et al., 2002].

5.2.1 Next location prediction

Some methods focus on the next location prediction problem, which is a relaxed version of trajectory prediction [Chen et al., 2014; Do et al., 2015; Gidófalvi and Dong, 2012; Jeung et al., 2010; 2008; Scellato et al., 2011]. The next location prediction methods can be classified into two groups according to their prediction method. The first group predicts the location of the user after a specific time, Δ, which is specified in each study. For example, Δ is 10 minutes and one hour in studies [Do and Gatica-Perez, 2014] and [Song et al., 2010] respectively. Sadilek et al. proposed a model for long-term prediction up to multiple years [Sadilek and

Krumm, 2012]. In some studies, the effect of Δ on performance is investigated [Scellato et al., 2011; Jeung et al., 2010]. Do et al. changed Δ to predict a set of locations visited by the user [Do et al., 2015]. The second group uses methods that do not take into account the time elapsed, but just focus on the next location that the user goes to, whether it is after a short or a long time [Gambs et al., 2012; Gidófalvi and Dong, 2012]. In both groups, the GPS coordinates are quantized into cell grid, and the output of the prediction is a cell rather than GPS coordinates.

Some approaches take advantage of other contextual factors to improve the performance [Cho et al., 2011; De Domenico et al., 2013]. For example, Domenico et al. improve the accuracy of the prediction by considering traces of multiple users and show the correlation between the trajectories which can be a signal of social interaction [De Domenico et al., 2013]. Next location prediction approaches mainly focus on making predictions at fixed and short time-scales, while our approach predicts the entire remainder of a person's day.

5.2.2 Machine learning models

Predicting one location among a set of finite locations (e.g., POIs, cells in a gridded map) makes the trajectory prediction problem similar to a classification problem. In this case, the locations are considered as the classes, and machine learning classification techniques are used for the next location prediction [Anagnostopoulos et al., 2011; Krumm and Horvitz, 2006; Tran et al., 2012]. Krumm et al. propose Predestination to predict drivers' destinations by producing a probabilistic map of destinations via Bayesian inference [Krumm and Horvitz, 2006]. Anagnostopoulos et al. consider visited locations as the feature vector, and then evaluate three classification methods [Anagnostopoulos et al., 2011]. Tran et al. extract the semantics of the visited locations and use them as the features for a classification tree [Tran et al., 2012]. To predict the next location within a smart building, Petzold et al. evaluate five machine learning approaches including Dynamic Bayesian Network, Multi-layer Perceptron, Elman net, Markov predictor, and State predictor [Petzold et al., 2006]. In [Do and Gatica-Perez, 2014], the performances of machine learning models such as Random Forest, Linear Regression and Logistic Regression for predicting 10 semantic location labels are compared in both personalized and user-independent modes. The more relaxed problem is occupancy detection, in which the number of the classes is two [Krumm and Brush, 2011; Scott et al., 2011]. Classification models cannot be applied to GPS data to predict coordinates that consist of continuous values. On the other hand, our method is able to predict the continuous values such as the latitude and longitude of the locations.

5.2.3 Markov models

In the literature, one of the common approaches for addressing the location prediction problem is the Markov model [Asahara et al., 2011; Gidófalvi and Dong, 2012; Chen et al., 2014; Gambs et al., 2012; Mathew et al., 2012; Do and Gatica-Perez, 2014]. The difference between Markov-based approaches is the way that the states are defined. For example, for GSM data, the cell towers are considered as states, while for WiFi data, WiFi access points are Markov model states [Zhao et al., 2011]. In a smart environment, proximity to a sensor can be considered as a state [Petzold et al., 2006]. Unlike those using fixed sensors, approaches using GPS data apply a prior spatial discretization (i.e. vector quantization). Some approaches simply use fixed grid on the spatial space [Morzy, 2007; Monreale et al., 2009; Chen et al., 2010], while others extract significant locations by clustering spatially or temporally [Lei et al., 2011; Ashbrook and Starner, 2003; Chen et al., 2014]. For example, Ashbrook et al. first, cluster GPS data points to find significant locations and then use Markov model to predict the user's movement from one significant location to another [Ashbrook and Starner, 2003]. Banovic et. al use a Markov Decision Process (MDP) framework to model routine behavior described as the user's daily commute. In the first step, the location logs, including latitudes, longitudes, and timestamps, are converted into states and actions representing the user's mobility for each day. The states indicate the day of the week, the hour of the day (0-24), the location of the user, and whether the user left, arrived, or stayed at the location for the past hour. The state transition probabilities are modeled with a stochastic MDP to consider the environment's influence on arrival time (e.g. travel distance, traffic) [Banovic et al., 2016].

Using Markov models for the trajectory prediction problem has two weaknesses. First, only the last location visited by the user is taken into account for the prediction of the next location. Some approaches use different orders of Markov models to obtain better accuracy [Chen et al., 2014; Ashbrook and Starner, 2003]. Second, the output of a Markov model is limited to the states representing a significant location, point of interest, CID, or any labelled location. Therefore, the Markov model is not applicable to GPS data unless a prior spatial discretization is applied (e.g. using a gridded map), and this reduces the granularity of the predicted locations. The discretization stage may use density-based clustering techniques to detect significant locations, stay points, or points of interests (POI) [Morzy, 2007; Chen et al., 2010]. The discretization, on one hand, reduces the complexity of the problem and increases the certainty of the results. On the other hand, it also reduces the precision of the approach due to the coarse granularity of the regions. As a result, predicted locations could include several important locations which are common in crowded cities where buildings are in close proximity to each other. For example, if the office and home are located in the predicted region, the location-based service cannot distinguish when the user goes to the office. Compared to the Markov models used for location prediction, our method can be used to predict the next transition time in the trajectory and the location at any particular time. Furthermore, there is no need for discretization in our method, which results in fine granularity.

5.2.4 Summary of gaps

This work addresses human mobility prediction across a day. Existing approaches do not make use of the rich information contained in the previous part of a user's daily trajectory. Furthermore, most of the current methods have been designed for discrete data, while our method can handle both continuous and discrete data (i.e. geographical and labelled trajectories). Unlike the other methods that predict the destination only, our method predicts the trajectory to the destination using spatio-temporal points. We believe this is vital for an effective location-based service. With trajectory prediction, we investigate not only where will the user be in the future but also how the user gets to that location and when. Specifically, our approach provides spatio-temporal points that can be either geographical coordinates or labelled locations. By obtaining this information, the user can be notified about the consequence of her movements in advance. In this way, a location-based system alerts the user about events that may happen during the trip to the destination.

5.3 Problem definition

In this section, after introducing the notations, we define the problem. Table 5.2 lists the symbols used in this chapter.

Definition 1: A *trajectory* is a trace of locations, represented by a series of chronologically ordered points, $p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_m$, where each point consists of a location and a timestamp $p = (loc, t)$. Thus, $< p_1, \cdots, p_i, \cdots, p_m >=< (loc_1, t_1), \cdots, (loc_i, t_i), \cdots, (loc_m, t_m) >$. For geographical trajectories, such as for GPS data, *loc* is a geospatial coordinate set; in labelled trajectories, e.g. a sequence of Cell IDs of WiFi Access points, *loc* is the label assigned

Table 5.2: Definitions of symbols

HT	set of historical trajectories
i	index of trajectory in HT
n	number of historical trajectories
T	time when the prediction starts
Tr	trajectory of the current day
Tr_{pre}	given sub-trajectory of the current day from 00:00 to T
Tr_{post}	to be predicted sub-trajectory of the current day from T to 23:59
Tr^i_{pre}	the first sub-trajectory of the i-th day from 0:00 to T
Tr^i_{post}	the second sub-trajectory of the i-th day from T to 23:59
$W(Tr^i)$	weight between Tr_{pre} and $Tr^i_{pre} \in HT$
μ	mean value
σ	variance

to a location.

Definition 2: The historical trajectories of the user containing n days of data is $HT = \langle Tr^1, ..., Tr^n \rangle$ where Tr^i denotes the trajectory of the user during the i-th day. A day starts at 0:00 and ends at 23:59.

Definition 3: *Current day* is the target day when the prediction takes place. The trajectory of the current day is Tr.

Definition 4: *Prediction time* (T) is the time of the current day when the prediction starts.

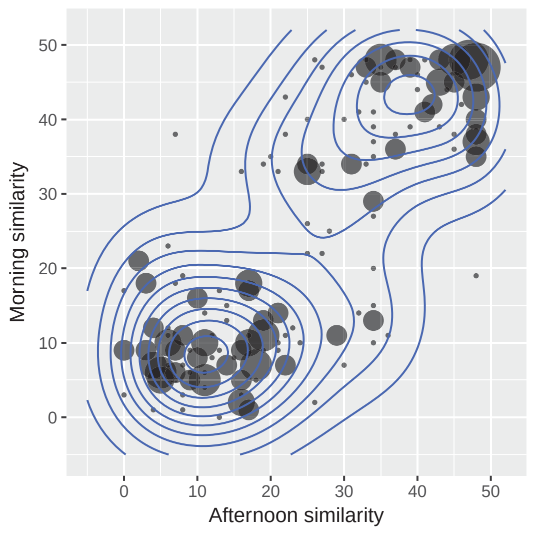
Definition 5: Tr^i_{pre} is the first sub-trajectory of the i-th day that is from $0:00$ to T while Tr^i_{post} is the second sub-trajectory of i-th day that is from T to $23:59$. For the current day, the sub-trajectory from $0:00$ to T is Tr_{pre} and the sub-trajectory from T to $23:59$ is Tr_{post}.

The definitions of the symbols are listed in Table 5.2. Given these definitions, we can now define our problem statement.

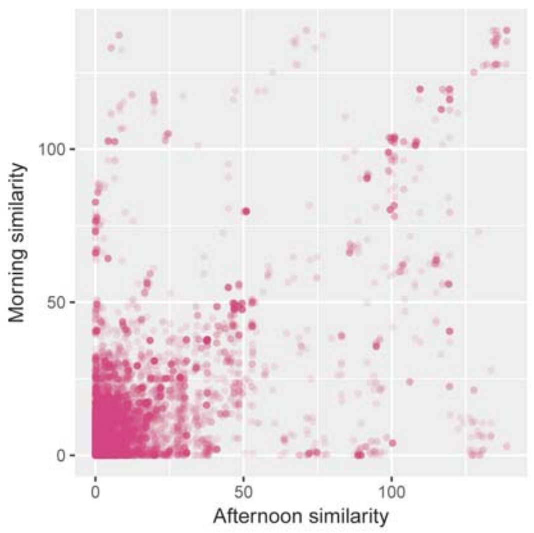
Problem: Given the historical trajectory, HT, and the user's initial trajectory of the current day, Tr_{pre}, our problem is the prediction of the trajectory for the rest of the day, Tr_{post}.

5.4 Observations in human daily trajectory

In this section, we discuss the rationale behind our approach by discussing three observations, each of which is examined on data described in the following subsection.

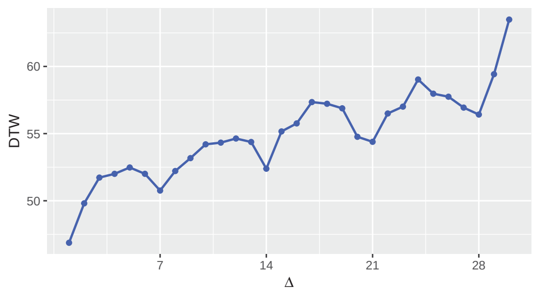


(a) Device Analyser dataset

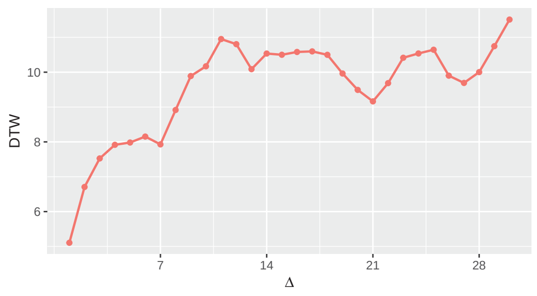


(b) MDC dataset

Figure 5.1: Correlation between the trajectories in the mornings and in the afternoons



(a) Device Analyser dataset



(b) MDC dataset

Figure 5.2: Temporal correlation between the trajectories that have Δ days difference

5.4.1 Data description

To validate the observations, we evaluate two types of datasets: a labelled dataset (Device Analyser) and a geographical dataset (Mobile Data Challenge). We use these same two datasets for our subsequent experiments.

- **Device Analyser**: The Device Analyser app gathers data about running background processes, wireless connectivity, GSM communication, and some system status and parameters. In this dataset, MAC addresses, WiFi SSIDs, and other forms of identification are hashed due to privacy purposes. Therefore, there is no ground truth or information about the geography or semantics of the locations. The trajectories consist of labelled cell tower IDs (CID), and the sampling rate is every 15 minutes. This makes the trajectory length equal to 96 for one day. In our experiments, we consider 225 users who have more than 40 days of data. For each user, we apply our approach to the last 10 days [Wagner et al., 2014].
- **MDC dataset**: The Mobile Data Challenge (MDC) dataset provides geolocation information for nearly 200 users. In addition to GPS data, WLAN data is also used for inferring user location. The location of WLAN access points was computed by matching WLAN traces with GPS traces during the data collection campaign. As an adequate amount of data is needed for prediction, we exclude users who do not have enough data and only considered users with more than 40 days of data. 136 users satisfy this condition. Similar to Device Analyser dataset, we process the same amount of data (i.e. 10 days) for each user and RMSE is reported over all instances [Kiukkonen et al., 2010].

The Device Analyser data provides hashed cell tower IDs (CIDs) and we call it the *labelled* trajectory dataset trajectory, composed of CIDs and timestamps. The MDC dataset provides the geographical location of the user and the trajectories include latitude, longitude, and timestamps and we call it the *geographical* trajectory dataset. When T is $12:00$, the trajectory of a day is split into morning and afternoon sub-trajectories.

5.4.2 Observation 1: Positive correlation between the morning and afternoon sub-trajectories

People's morning trajectories are positively correlated with their afternoon trajectories. That is to say, if a user has the same trajectories over two mornings, it is highly likely that she will have the similar trajectories in the afternoons.

Figure 5.1 shows the scatter plot for similarities in the mornings and in the afternoons measured by Dynamic Time Warping (DTW). (We explain later how we apply DTW to our labelled and geographic datasets.) In Figure 5.1a, each of the 2000 points represents 2 random days taken from the same user in the labelled dataset. The vertical axis represents the morning DTW difference of the two days, and the horizontal axis represents the afternoon DTW difference, where a smaller difference indicates higher similarity. Since this is labelled data, the DTW differences are integers ranging from 0 (high similarity) to 48 (high difference). The strongest cluster in this plot is centered around (10,10). This indicates that days that are relatively similar in the morning and also relatively similar in the afternoon. Exact matches are rare, hence there are no points at (0,0). The size of each point is proportional to the number of points occupying that coordinate.

Figure 5.1b shows the same analysis for 2000 random pairs of days in the geographic dataset, where the DTW difference is a continuous value. We again see a large cluster near (0,0). Taken together, the plots in Figure 5.1 imply that if two trajectories are similar in the morning, it is expected that they are similar in the afternoon, too. This validates the first observation. For example, if someone works for two companies and has two routines, the morning trajectory often identifies which routine will be followed in the afternoon. We make use of this observation by predicting a person's afternoon trajectory based on their morning trajectory

5.4.3 Observation 2: Positive temporal correlation

The importance of each historical trajectory varies and depends on its date. To show the date effect, 2000 random trajectory pairs are selected from each dataset. Each trajectory pair includes two trajectories from two days with less than 30 days gap from the same user. Figure 5.2 shows how much the closeness of the dates reflects the similarity between two trajectories, where similarity is again measured by DTW. Specifically, the x-axis indicates the gaps (Δ) in days between the dates of two trajectories, while the y-axis reflects the average similarity. It is observed that 1) overall, as Δ increases, the similarity decreases, and 2) when $\Delta = 7$, the weekly periodicity appears, which means trajectories are more similar between two days with 7 days difference. Thus, in general, the closer the date of a historical trajectory to the current prediction date, the more important it is. The reason is that people often have a similar routine in two close days (e.g. two consecutive days) [Cho et al., 2011]. This temporal correlation indicates that higher priority should be given to the trajectories with the closer dates or dates that are one week apart. For example, for prediction of Monday mobility, this observation implies that it is better to use the trajectory of the last Monday rather than the trajectory from the last Tuesday. We use the temporal correlation to compute weights on each previous day to use for predicting the current day. Days with higher correlation are given more weight.

5.4.4 Observation 3: Outlier trajectories

There are outlier trajectories, which are different from people's normal routine (i.e. visit to a location by exception) and which are unlikely to happen again (especially considering the entire trajectory as a whole). Then, when making trajectory predictions, these outlier trajectories should be excluded. Here, we elaborate one example to clarify the effect of outlier removal stage. Assume that the user has a strict routine on Wednesdays and we want to predict the afternoon trajectory given a Wednesday morning trajectory. The temporal correlation specifies to pick the afternoon trajectory of the last Wednesday because it is the closest day to the current day with similar morning trajectory. Now, assume that on Wednesday afternoon last week, the user deviated from his routine to visit a new location (e.g. visiting a friend in a hospital) and this visit had never been repeated in the historical data. Observation 3 implies that such a visit was temporary and unlikely to happen again. In other words, the predicted trajectory should not be nor include an outlier.

In summary, the proposed method is based on the above observations. 1) We predict Tr_{post} using the historical trajectories that have the sub-trajectories similar to Tr_{pre}. 2) We calculate temporal correlation to give priority the historical trajectories based on their relative date. 3) The historical trajectories that include outliers are not used for the prediction.

5.5 Partial human daily trajectory prediction

In this section, we provide an overview of the proposed approach to partial human daily trajectory prediction, as shown in Figure 5.3:

(a) The first stage is to compare Tr_{pre} with Tr^{i}_{pre}, the morning sub-trajectories from the historical trajectories. The goal is to weight the historical trajectories based on their similarities to Tr_{pre}. Two similarity metrics are deployed in our approach: Dynamic Time Warping (DTW) [Keogh and Pazzani, 2000] and Edit Distance(ED) [Chen et al., 2005]. We will describe these metrics later in the section.

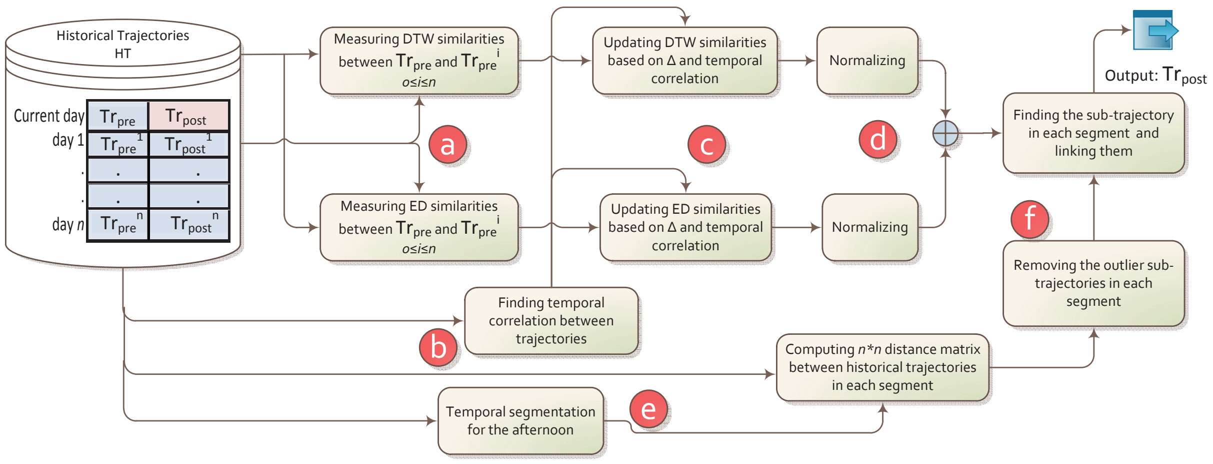

Historical Trajectories HT
Current day
day 1
day n
Tr_pre
Tr_post
Measuring DTW similarities between Tr_pre and Tr_pre^i 0≤i≤n
Measuring ED similarities between Tr_pre and Tr_pre^i 0≤i≤n
Updating DTW similarities based on Δ and temporal correlation
Updating ED similarities based on Δ and temporal correlation
Normalizing
Normalizing
Finding temporal correlation between trajectories
Computing n*n distance matrix between historical trajectories in each segment
Temporal segmentation for the afternoon
Removing the outlier sub-trajectories in each segment
Finding the sub-trajectory in each segment and linking them
Output: Tr_post
a
b
c
d
e
f

Figure 5.3: Diagram of the approach to partial human daily trajectory prediction

(b) The temporal correlations are processed to quantify the impact of date differences on the prediction.

(c) Together with (a), the temporal correlation between Tr_{pre} and Tr^i_{pre} are used to enhance the corresponding importance of Tr^i_{pre}. For example, the afternoon trajectory of yesterday might receive a higher weight compared to the afternoon trajectory of a day from last month.

(d) The two similarity measures (DTW and ED) are normalized in this stage. After normalizing the similarity measures, they are combined to form the final weight of each historical trajectory. The final weights indicate the similarity of the morning parts of the historical trajectories to Tr_{pre}. The afternoon part of the trajectories that received high weights will be used for prediction in the final stage.

(e) Information Gain Temporal Segmentation (IGTS), introduced in Chapter 3, is deployed on the historical afternoon trajectories. Then, the prediction is performed within each segment. Ideally, each temporal segment represents the period over which an activity is undertaken. For example, if the user often goes for lunch between 12:00 and 13:00, [12:00, 13:00] is one of the temporal segments.

(f) In each afternoon segment, a distance matrix is built. The distance matrix indicates the distance/similarity between two sub-trajectories in the historical data during that segment. Based on the distance matrix, we discard the outlier sub-trajectories that represent the trajectories that are unlikely to happen during the current day. From the

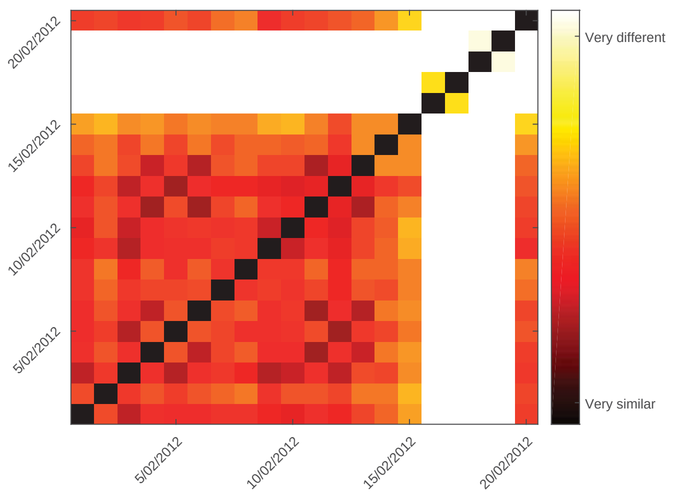


Figure 5.4: Similarity matrix for 20 successive days. The darker the cell, the stronger the similarity.

rest of the trajectories, we choose the sub-trajectory of a day with the highest weight assigned in stage (d). The predicted sub-trajectory are linked together to form Tr_{post}.

We detail each of the above components in the following sections, including trajectory similarity metrics, temporal correlation, temporal segmentation, and outlier removal.

5.5.1 Similarity metric

There are several metrics to measure the similarity between two trajectories. For example, the four most common metrics are Fréchet Distance, Dynamic Time Warping (DTW), Longest Common Sub-Sequence (LCSS), and Edit Distance (ED), all of which have been introduced and compared by Toohey and M. Duckh [Toohey and Duckham, 2015]. Of these metrics, DTW relies on matching points in trajectories. Specifically, a single point in one trajectory can be matched to multiple points on the other trajectory based on the distances. The calculation of distances between the points is performed using a chosen distance function. For labelled trajectories, the distance function only checks whether the two labels are equal or not. If the two labels match, the distance is zero. Mismatched labels have a distance of one. For geographic trajectories, the distance between points is the great circle distance measured from the two latitude/longitude pairs. DTW considers delays in the trajectories. This means that similar sub-trajectories are matched together even though their timestamps are different. However, outliers can significantly affect DTW because there has to be a match between every point in both trajectories [Toohey and Duckham, 2015]. ED aims to count the minimum number of edits needed to make two trajectories equivalent. This means after the edits all of the locations in the trajectories have to be the same. For geographical trajectories (i.e. MDC dataset), we consider two locations as the same location if they are within 0.1 range in latitude and longitude. Among different variations of edit distance, we use the one described by Chen et al. [Chen et al., 2005]. ED compares the trajectories in every fine-grained time slot and does not consider delays in the trajectories. ED is more robust in the treatment of outliers than DTW.

As the similarity metric plays a critical role in our approach, we propose a metric that integrates DTW and ED together to provide a comprehensive similarity measure. This is reasonable, because 1) Fréchet distance and DTW are highly correlated ($R = 95\%$); 2) LCSS and ED are highly correlated ($R = 83\%$) [Toohey and Duckham, 2015]. Furthermore, DTW and ED are compatible with non-geographic (labelled) trajectories such as trajectories represented by cell tower IDs.

Figure 5.4 shows the similarity matrix over 20 successive days using DTW based on the Device Analyser (labelled) dataset. Specifically, each entry denotes the similarity value between trajectories in two days. It is observed that days 17-20 are very different from the other days. The user may have gone for a trip in that period to visit new places in other cities. And the calendar shows that this period is from Friday to Sunday, which strengthens this hypothesis.

5.5.2 Weighting historical trajectories

Here, we specify how we combine DWT and ED with the observed temporal correlation between trajectories. Given Tr_{pre}, its DTW similarity to $Tr^i_{pre} \in HT$ is defined as follows:

$$W_{DTW}(Tr^i) = DTW(Tr_{pre}, Tr^i_{pre}) \times T^{DTW}_{cor}(\Delta), \tag{5.1}$$

where T_{cor} denotes the effect of the temporal correlation in the weighting. Δ denotes the difference between the dates when Tr_{pre} and Tr^i happened. To calculate $T_{cor}(\Delta)$, we use the historical trajectories to find the correlation between the trajectories with Δ days difference. Here, we are emphasizing the effects of Observation 2 that Δ is an important factor governing the similarity between days based on how far apart they are in time." Similarly, the ED simialrity is defined as follows:

$$W_{ED}(Tr^i) = ED(Tr_{pre}, Tr^i_{pre}) \times T^{ED}_{cor}(\Delta). \tag{5.2}$$

Small values of $W_{DTW}(Tr^i)$ and $W_{ED}(Tr^i)$ mean that the Tr_{pre} and Tr^i_{pre} are similar. According to Observation 1, among the historical trajectories, ones that are most similar to Tr_{pre} in the morning should get higher weights when predicting the afternoon trajectory.

Before summing up the ED and DTW similarity values, they are z-normalized. That is, each type of similarity values has zero mean and a variance of one. This provides a comprehensive similarity measure that plays a critical role in our approach.

$$W(Tr^i) = \frac{W_{DTW}(Tr^i) - \mu_{DTW}}{\sigma_{DTW}} + \frac{W_{ED}(Tr^i) - \mu_{ED}}{\sigma_{ED}}, \tag{5.3}$$

where μ and σ denotes the mean and variance that are calculated for each user separately, and they are based on the user's historical data. $W()$ is the total weight assigned to a historical trajectory considering ED and DTW metrics and temporal correlations. $\frac{W_{DTW}(Tr^i)-\mu_{DTW}}{\sigma_{DTW}}$ and $\frac{W_{ED}(Tr^i)-\mu_{ED}}{\sigma_{ED}}$ are the normalization for DTW and ED, respectively. This normalization is introduced to remove the scale effects and gives both metrics an equal priority.

5.5.3 Temporal segmentation

Factorizing a time period into several temporally homogeneous segments is called temporal segmentation. Considering trajectories from a user, temporal segmentation reveals the departure times when the user changes her activities. For example, assume one leaves home at 7 am, works at the office from 9 am to 4 pm, and then goes outdoors until 6 pm every day. In this example, 7am, 9am, 4pm, and 6pm are the departure times and [7am-9am], [9am-4pm], [4pm-6pm], and [6pm-7am] are the temporal segments .

To predict the trajectory of the user in the afternoon, Information Gain Temporal Segmentation (IGTS), introduced in Chapter 3, is applied to historical trajectories to find the usual changes in the user's daily activities, such as when the user usually goes from home to work. Then, a sub-trajectory is predicted for each segment. Temporal segmentation allows us to analyse fine-grained segments and discard the outlier sub-trajectories in each segment.

As discussed in Chapter 3, IGTS computes the distribution of the user's locations in each segment and tries to capture the segments that have the lowest entropy. Low-entropy segments imply that the user's location is predictable in that segment. For example, [12am-

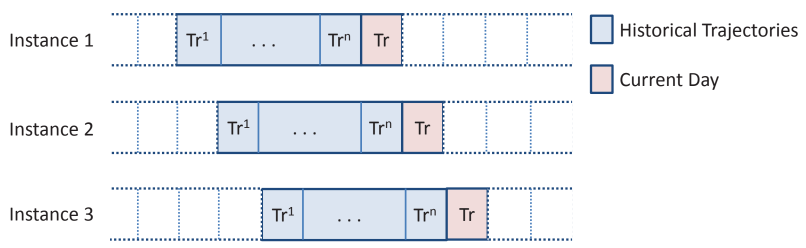


Figure 5.5: Three sample instances from one user. Each block represents one day.

6am] is a low-entropy segment for a common user because the user is probably at home during this period. If the trajectories contain geographic locations, before using IGTS, the locations should be quantified (e.g. by using gridded map). To find the number of the segments, IGTS uses a formula to choose the best candidate from a range of numbers based on knee-point detection. If the number of the segments is too large, the predicted trajectory will have a high variation. Therefore, we select the best candidate for the number of segments ranging from 2 to 6.

5.5.4 Outlier removal

In this stage, we discard the historical trajectories that have abnormal sub-trajectories in the afternoon. According to Observation 3, these trajectories are discarded because it is unlikely to have a sub-trajectory similar to the abnormal sub-trajectories. The abnormal sub-trajectories are sub-trajectories that are not found in the historical data such as going to the airport to pick up someone, visiting a friend in a hospital, or inspecting an apartment to buy. The outlier removal stage is done regardless of similarities in the morning parts. This means we discard the historical trajectories with abnormal sub-trajectories in the afternoon even if the morning parts are similar to the current day.

To remove the outliers, we build a distance matrix for each afternoon segment results from temporal segmentation stage (e.g. from 1 pm to 4 pm). The distance matrix, D^{S1}, is an $n \times n$ matrix where D^{S1}_{ij} denotes the distances between the sub-trajectories i and j in segment S_1. We use DTW to compute the distance matrix. An outlier sub-trajectory is based on its closest neighbour (the most similar sub-trajectory). If the closest distance is higher than a threshold, the sub-trajectory is considered as an outlier. The outlier removal is performed in each segment independently.

5.5.5 Linking trajectories

Finally, we make the prediction for each segment obtained from the component of temporal segmentation, then link these predictions together to form the prediction of Tr_{post}. Namely, the corresponding segment in $Tr^i_{post} \in HT$ with the highest weight $W(Tr^i)$ is selected as the prediction for each segment in Tr_{post}. Then, we naturally link the predictions for each segment together to make the prediction for Tr_{post}.

5.6 Experiments

Before evaluating the effectiveness of the proposed method thoroughly, we clarify the experiment setting. For each user, we pick m consecutive days of trajectories ($m > n$), where n denotes the size of historical trajectories used for the prediction of one day afternoon trajectory. For each day, we run our approach while considering the past n days as the historical data, using the (n+1)-th day as the first test trajectory. Based on the morning trajectory (i.e. the given part of the trajectory of the current day), we predict the afternoon part of the (n+1)-th day. For the next day, again, we predict the afternoon trajectory based on the past n days and the morning trajectory of that day. We continue the prediction of the afternoon sub-trajectories for each day until we reach the m-th day. Therefore, we have $(m - n)$ test instances for each user. In other words, each day of data is treated as a test instance. Figure 5.5 shows three sample instances from one sample users' trajectories.

For each instance, the predicted trajectory is compared to the actual trajectory and the error is calculated using DTW. The prediction error shows how much the predicted trajectory is similar to the actual trajectory. If the predicted trajectory is exactly the same as the actual trajectory, DTW returns zero. Otherwise, the error is the DTW distance between the two trajectories. DTW is an effective way to evaluate the prediction, because it accounts for both the spatial and temporal differences between the prediction and the actual trajectory. Calculating the error for each instance, the Root Mean Square Error (RMSE) is reported for all data. Specifically, RMSE is the root mean square of the prediction errors measured by DTW. For the labelled trajectories, we also report precision and recall in addition to RMSE. However, it should be noted that precision and recall only consider the set of the predicted locations and do not consider the sequence and transition times. In the experiment results, our method is denoted by "TrAf".

5.6.1 Baselines

For comparison, we use the following baselines:

- **Last week trajectory**: According to Section 5.4.3, of the historical trajectories, one of the most similar trajectory to Tr_{post} is the trajectory of the afternoon of the last week. We use this as our first baseline. This baseline is similar to "Same place" baseline used in the next location prediction methods [Do et al., 2015].
- **Most Frequent visit**: This baseline finds the most visited locations at each time of the day. For example, for finding the location at 1 pm, the method searches for the location in the historical trajectories that is most often visited at 1 pm [Do et al., 2015; Do and Gatica-Perez, 2014].
- **PreHeat**: This algorithm was designed for the problem of occupancy detection. Specifically, given the historical records and the morning occupancy, PreHeat predicts the occupancy in the afternoon. To this end, this method detects 5 days that have the most similar occupancy patterns to the current day in the morning and then, the probability of the occupation is calculated based on the detected days. We adapt PreHeat to make it applicable to our data. For labelled trajectories, we consider one day instead of 5 days because it is not possible to calculate the mean value of labels. For geographical trajectories, the mean values of the latitude and longitude are considered as the coordinate of the predicted location. We also evaluate the algorithm when the number of the detected similar days is 10. [Scott et al., 2011].
- **Markov model**: It is one of the most popular approaches for mobility prediction problem [Asahara et al., 2011; Gidófalvi and Dong, 2012; Chen et al., 2014; Gambs et al., 2012; Mathew et al., 2012; Do and Gatica-Perez, 2014]. However, Markov models cannot be applied directly to the GPS data and there should be a discretization stage for defining Markov states. In our implementation, we use a grid map for discretization, and each cell is considered as a state.

5.6.2 Experiments on labelled trajectories

In this experiment, we predict the labelled afternoon trajectories of 2250 days from 225 Device Analyser users (10 for each user). The length of the historical data is 30 days, i.e. $n = 30$, which means for prediction of the afternoon trajectory, Tr_{post}, we process the morning

Table 5.3: Tests show that our algorithm's mean performance is better than the baseline algorithms on the Device Analyser dataset by a statistically significant margin

	TrAf	PreHeat	Most frequent visit	Last week trajectory
RMSE	**25.6**	26.1	28.5	31.4
Wilcoxon test p-value	-	0.0015	0.00001	< 0.0001

trajectory, Tr_{pre}, as well as the trajectories of the user during the previous 30 days. Here, T is 12 pm which means we have the trajectory of the current day up to 12 pm and trajectory between 12 pm and 12 am is unknown. This makes the lengths of Tr_{pre} and Tr_{post} equal to 48. The impacts of the prediction time and size of historical trajectories are investigated in the following subsections.

In Figure 5.6, the y-axis denotes the similarity between the predicted trajectory and the ground truth. The box plot containing inter-quartile range (IQR), a measure in descriptive statistics. The IQR is the 1st quartile (25%) subtracted from the 3rd quartile (75%). The box demonstrates the values between 1st quartile and 3rd quartile while whiskers extend to data within 1.5 times the IQR. The median is also shown in the box. The maximum error is 48 and this happens when none of the predicted Cell tower IDs (CIDs) is the same as the actual CIDs. The figure shows that the trajectories predicted by our method are more similar to the actual trajectories. To show that this superiority is not by chance, a statistical analysis, Wilcoxon test, is conducted. RMSE and the p-values are reported in Table 5.3. All the tests show our algorithm's mean performance is better than the baselines' means to the 0.0001 significance level.

It can be seen that the variation of the result is high and the certainty of all methods are low. The reason is that the human mobility has some limits for prediction and it is not completely predictable [Song et al., 2010]. In fact, it is likely to see new trajectories that are different from the historical trajectories. For example, the user may leave the office earlier, or she may catch up with her friends after the working hours.

The reality is that the performance of each method depends on the user's behaviour. For example, if morning trajectories and afternoon trajectories of a user are highly correlated then our method works better. For users that have strict routine over a week, the trajectory of the last week is an appropriate estimation for today's trajectory compared to other methods. However, our method has the best result on average.

In addition to DTW, which considers temporal and spatial aspects at the same time, we

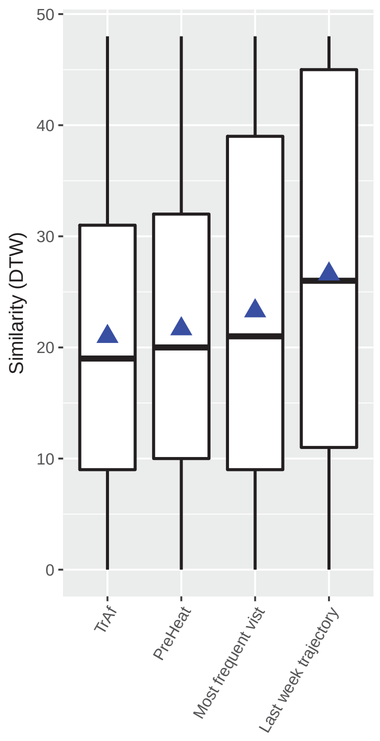


Figure 5.6: Results for 2250 instances from the Device Analyser(labelled) dataset. ▲ denotes the mean value.

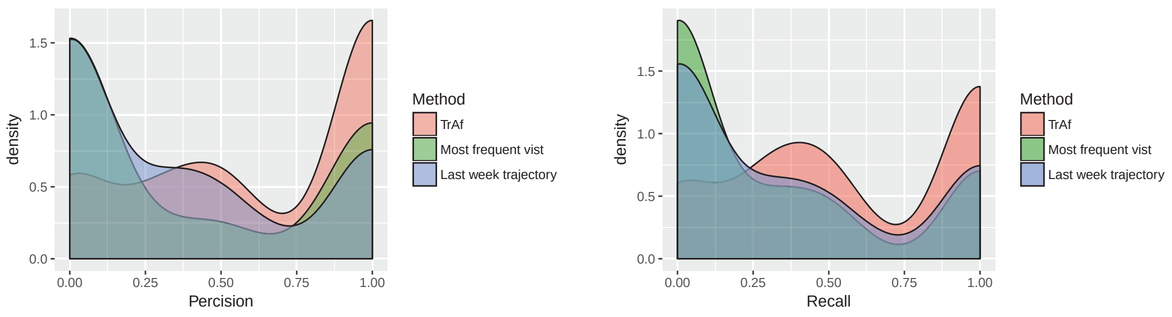


Figure 5.7: Distribution of precision and recall of predicting the full set of locations of the whole day (Device Analyser dataset). The temporal aspects of the errors are not considered in this experiment.

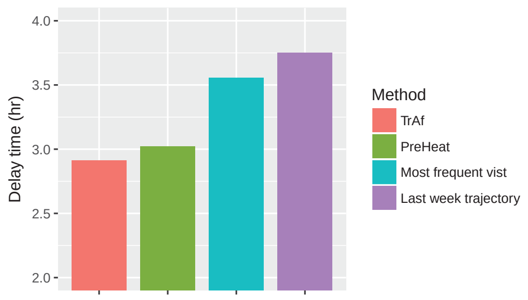


Figure 5.8: Average error in predicting the departure times (Device Analyser dataset). The spatial aspects of the errors are not considered in this experiment.

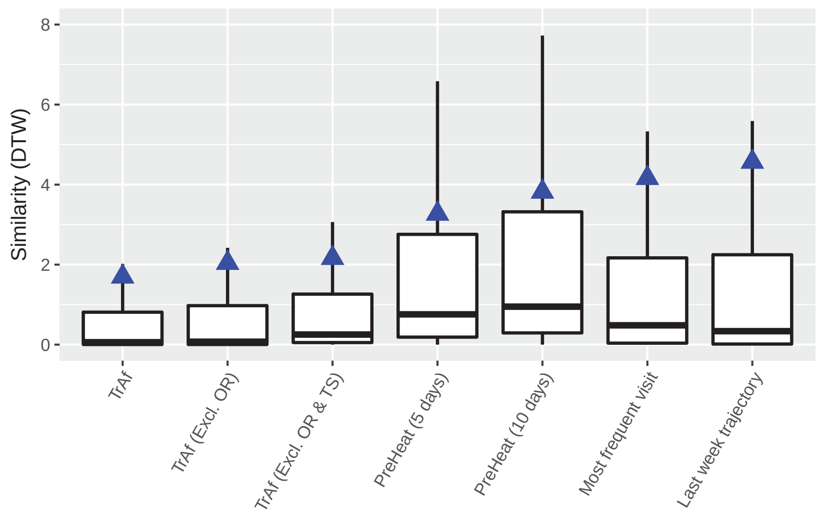
conduct temporal and spatial evaluation separately. For spatial evaluation, we report the distribution of precision and recall in Figure 5.7. To calculate precision and recall, the set of predicted locations is compared to the actual locations visited by the user. Clearly, the temporal aspects such departure time are not considered. For temporal evaluation, we report the difference between the actual and predicted first departure time after 12 p.m. Figure 5.8 shows the results. It can be seen that our method outperforms the baselines in both experiments.

5.6.3 Experiments on geographic trajectories

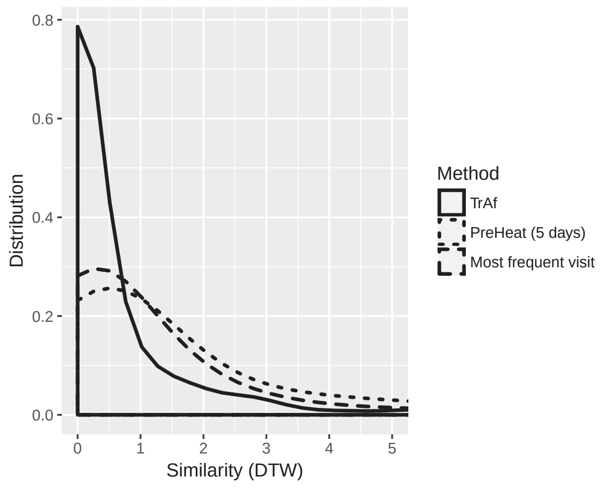
From the MDC dataset, all the users with more than 40 days data are chosen for evaluation (136 users). The size of the historical trajectory is 30 days in this experiment, i.e. $n = 30$. The total number of test instances is 1360 (10 for each user). For each instance, we measure the similarity between the predicted trajectory and the ground truth.

Figure 5.9(a) shows the box plot. For each method, the ▲ indicates the mean value while the box plots report inter-quartile ranges (IQR). To show the impact of Temporal Segmentation (TS) and Outlier Removal (OR), we run our method with and without these stages. The results indicate the positive impact of TS and OR. Figure 5.9(b) shows the distributions of the DTW distance between the actual and predicted trajectories, where we can see that the trajectories returned by our method are closer to the ground truths with higher probabilities of low DTW.

The RMSE of our method is 35% less than the best baselines, which has a significant impact on the level of accuracy that location service recommendation can achieve and therefore on the user experience. Similar to other experiments, we evaluate the performance using



(a)



(b)

Figure 5.9: (a) Box plots for the baselines and different settings of our approach. y-axe shows the similarities between the predicted trajectories and ground truths. ▲: mean value, OR: Outlier Removal, TS: Temporal Segmentation (b) Distributions of the similarities for three methods

Table 5.4: Tests show that our algorithm's mean performance is better than the baseline algorithms on the MDC dataset by a statistically significant margin

	TrAf	TrAf (Excl. OR)	TrAf (Excl. OR & TS)	PreHeat (5 days)	PreHeat (10 days)	Most frequent visit	Last week trajectory
RMSE	**6.3**	7.5	7.6	8.6	10.1	13.0	14.7
Wilcoxon test p-value	-	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001

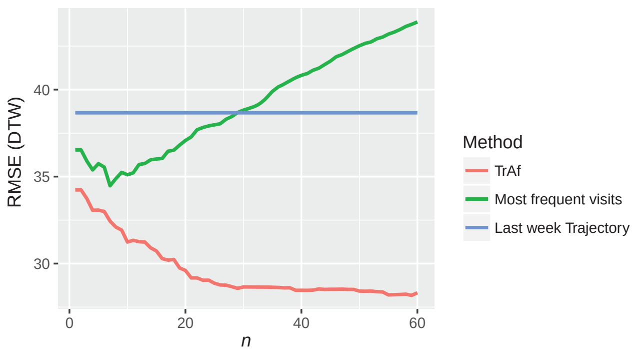


Figure 5.10: Effect of the historical trajectory size on the accuracy of prediction. The performance of our approach improves with increasing the historical trajectory size.

DTW, which is able to consider both the temporal and spatial deviation. 35% reduction in error means, on average, that the user is 35% closer to the predicted location or the user has to wait for 35% less time for an event to take place. In practical terms, this means that the user spatial and temporal proximity to a recommended *event* (e.g., a show, a train to catch) is more than doubled.

5.6.4 Impact of historical trajectory size

Here, we investigate the impact of the size of the historical trajectories, n, on the accuracy of the prediction. Specifically, we vary n from 2 to 60 days with the step length 1 on the Device Analyser (labelled) dataset. In this experiment, we use three users with a total of 943 days of data (467, 221, and 255 for the three users). The RMSE is reported in Figure 5.10.

From Figure 5.10, we can observe that: 1) the last week trajectory approach is not affected because it does not consider the historical trajectories; 2) increasing n makes the most frequent visits approach more accurate. This trend continues until $n = 7$, and after that increasing the historical trajectory size makes the approach worse. The reason is that the user may change behaviour and may not go to the places that she used to go. A long series of historical data causes the most frequent visits approach to promote the locations that

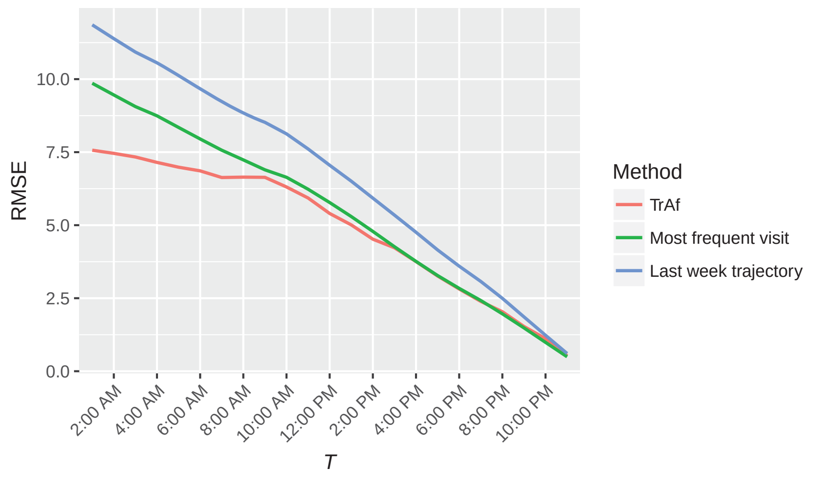


Figure 5.11: Impact of prediction time on the performance. Having more data from the current day leads to more accurate prediction.

have been frequently visited a long time ago; 3) the proposed method handles the size of the historical trajectories well. There are two reasons for this. First, by considering the temporal correlation, we give priority to the trajectories from consequential dates. Generally, closer dates are more consequential. This gives the trajectories from a long time ago less effect. Secondly, our method only uses the historical trajectories that have the similar morning trajectory as Tr_{pre}. Therefore, if the behaviour of the user changes, our method recognizes it by analyzing and comparing the morning trajectories.

Changing n causes a trade-off between the accuracy and processing time. For a large value of n, we have accurate results but at the same time, our approach becomes more costly. In our experiment, we set n to a number between 20 to 30, because increasing n more than 30 does not lead to a significant increase in accuracy.

5.6.5 Impact of prediction time

According to our problem formulation, the current day trajectory of the user up to prediction time T is known. Here, we investigate the impact of prediction time, T, on the accuracy. To this end, we design the experiments by varying T from 1 am to 11 pm with a step length of 1 hour, and we use the last 20 days as the historical trajectories, i.e. $n = 20$. In this experiment, the same set of users as Section 5.6.3 are analysed. Specifically, the experiment on the geographic dataset is repeated 23 times (once for each T). Figure 5.11 shows the relationship between prediction accuracy in RMSE and prediction time. It is observed that:

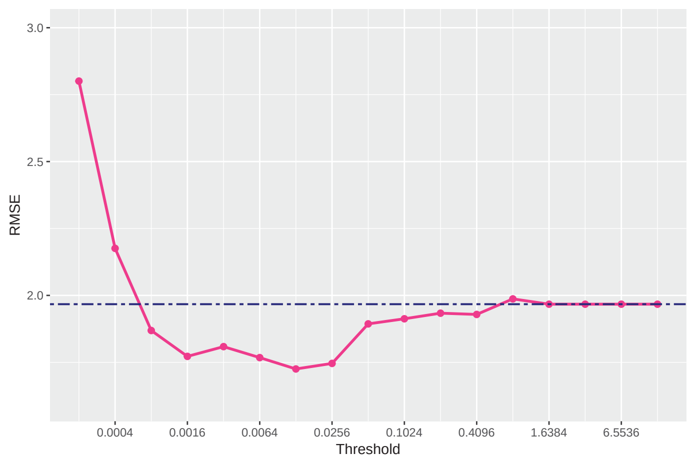


Figure 5.12: Impact of the threshold of the outlier removal on the accuracy of prediction. The dashed line shows when there is no outlier removal stage.

1) the trends are decreasing because DTW returns smaller values for short trajectories; 2) furthermore, when we have more data from the current day, the prediction is more accurate. If the prediction time is after 7 pm, our method and ”most frequent visits” overlap, otherwise our method outperforms the baselines; 3) there is a steeper decrease starting around 12 pm in the DTW of our method. This means the trajectories from 11 a.m. to 12 p.m. is very informative and more predictive.

5.6.6 Impact of outlier removal

In this experiment, the effect of the outlier removal stage with different thresholds is investigated. Figure 5.12 shows the impact of the threshold used for detecting outliers on the accuracy of the proposed approach on the geographic dataset. The x-axis has a logarithmic scale to assist interpretation. When the threshold is too small, a high portion of the sub-trajectories is recognized as outliers. In this case, the error is high because some useful information for the prediction is discarded. By increasing the threshold up to 0.0128, the results improve further. When the threshold increases, fewer sub-trajectories are recognized as outliers. When the threshold is higher than 1.638, none of the sub-trajectories are recognized as outliers and the performance is the same as the approach without outlier removal.

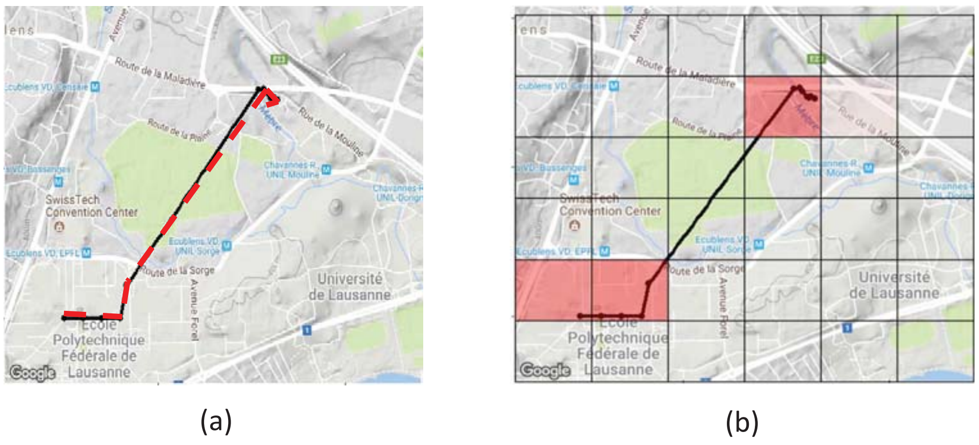


Figure 5.13: (a) Sample output of the proposed approach. The red dashed line shows the predicted trajectory of our method. The black line is the actual trajectory of the user (b) Sample output of the Markov model. The red cells in the gridded map show the prediction of the Markov model.

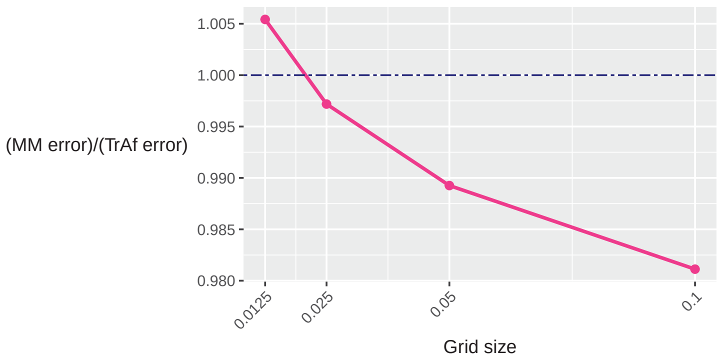


Figure 5.14: Comparison between our method and the Markov model for different grid size. The y-axis indicates the ratio between the error of our method and the Markov model. The blue dashed line shows when both approaches have the same performance.

5.6.7 Experiments on the gridded map

Here, we compare our method with a first-order Markov model. To run Markov model on the geographic dataset, we use a gridded map to discretize the GPS data. In this case, each state is one of the cells in the gridded map. Consequently, the locations predicted by Markov model are cells rather than geographic coordinates. Figure 5.13 shows the difference between the outputs of our method and the Markov model. To compare the results, we discretize the output trajectory from our method. Figure 5.14 shows the effect of the gridded size on the performances of both methods. For large grid sizes, the Markov Model performs better than our method. However, when the grid size is reduced, our method performs better, because the data becomes sparse and there is not enough historical data to build a Markov Model.

5.7 Conclusion

This chapter presents a method for completing the user's daily trajectory using the initial trajectory of the current day and historical trajectories. The algorithm takes both temporal and spatial aspects into account to investigate the similarities between the sub-trajectories. To improve the performance and reliability of the method, we add some other phases including temporal segmentation, extracting temporal correlation, and outlier removal. The method is applied to the situation where user trajectories are either labelled or geographical.

This chapter concentrates primarily on the issues of accuracy. We compare our method to four baselines with different parameters, and the results show that the proposed method significantly outperforms the baselines in terms of accuracy. The positive impacts of temporal segmentation and outlier removal stages are reported in the experiments. Furthermore, we investigate the impact of different parameters on our method including the prediction time, historical trajectory size and granularity. The experiment results clarify that the larger historical data contains more information and improves the accuracy of the prediction. Furthermore, having more data from the current day makes the prediction more accurate. The high level of precision obtained by the technique has the power to unlock more precise location service recommendations.

Chapter 6

Conclusion

Human mobility plays a significant role in numerous applications and therefore has become a core subject in multiple disciplines. In urban computing, understanding human mobility enhances services in smart cities. Social scientists have long been theorising with models that describe the mobility of individuals through a day. Similarly, ecologists use mobility models to explain resource consumption norms in ecosystems. These models are built based on specific types of data such as temporal, spatial or spatio-temporal data, depending on the data sources. In some cases, the mobility of a user is captured through pervasive temporal data such as sensor data, while in other cases the spatial aspect of their mobility is captured. Furthermore, the desired computational cost varies depending on the application. In some applications, due to resource restrictions, it is not feasible to process and store detailed locations.

In this dissertation, we have taken a step forward in both the abstract modelling of a human movement graph and the development of approaches that could analyse temporal and spatio-temporal pervasive signals. Although the proposed approaches are designed to mine pervasive signals, they are quite generic and we evaluate their application to other types of datasets too. For temporal segmentation, the proposed method has been applied to a device-free posture recognition dataset, Bluetooth movement detection dataset, and human activity dataset in addition to datasets that capture human mobility such as the Device Analyser connectivity dataset. For graph summarisation, Shrink has been applied to a friendship network, collaboration network, web graph and social network in addition to a movement graph and road network. Similarly, our trajectory prediction approach is generic and has been applied to both labelled and geographical trajectories.

6.1 Summary of contributions

The first question that should be clarified before tackling human mobility challenges is the type of data involved. In this thesis, we have proposed new approaches for each type of data separately, including temporal, spatial, and spatio-temporal data.

In Chapter 3, while blind to the locations, we have proposed an Information Gain-based Temporal Segmentation method (IGTS), an unsupervised segmentation technique, to find the transition times in human daily life, from heterogeneous sensor data. We have analysed time series to find times when the user changes their behaviour. In our method, there are no restrictions to the time series and they could be from any source with any frequency. No generic method has been proposed for extracting transition times at different levels of activity granularity. Existing work in human behaviour analysis and activity recognition has mainly focused on either at low-level, such as standing or walking, or high-level, such as dining or commuting to work. The proposed IGTS method is applicable for low-level activities, where each segment captures a single activity, such as walking, that is going to be recognized or predicted, and also for high-level activities. The heterogeneity of sensor data is dealt with a data transformation stage. The generic method has been thoroughly evaluated on a variety of labelled and unlabelled activity recognition and routine datasets from smartphones and device-free infrastructures. The experiment results demonstrate the robustness of the method, as all segments of low- and high-level activities can be captured from different datasets with minimum error and high computational efficiency.

In Chapter 4, we have introduced a new method for graph summarisation while preserving the distances within the graph. Applying Shrink to a movement graph that contains spatial information, we have changed the granularity of the locations. We have also investigated the impact of summarising on the accuracy and efficiency. To sum up,*Shrink* possesses the following features that make it practical for real-world applications: (1) it is linear in the number of nodes when $\sigma << |V|$, where $|V|$ is the number of nodes and σ is the average degree (see Section 4.6.2). This is common in large graphs with thousands or millions of nodes; (2) the larger the original graph is, the more accurate *Shrink* is. The reason is that large graphs usually have long paths and *Shrink* has less effect on the length of the long paths (see Section 4.5.2 and 4.8.4); (3) the error rate and the compression ratio are adjustable; (4) it provides not only distances but also the corresponding instances (nodes) of the shortest paths;(5) it is applicable to all types of distance queries, including reachability, single-source shortest-path (SSSP), all-pairs shortest path (APSP), closeness centrality and betweenness centrality. The experiment results show that compressing a two-million-node graph into fifths has the average error less than 1%.

In Chapter 5, we have predicted the user's trajectory by mining spatio-temporal data and leveraging similarities between trajectories in mornings and afternoons. Existing work on human mobility prediction has mainly focused on the prediction of the next location (or the set of locations) visited by the user, rather than on the prediction of the sequence of further locations and the corresponding arrival and departure times. However, in our method, the predicted trajectory includes the sequence of future locations, the staying times, and the departure times. Furthermore, existing approaches often return predicted locations as regions with coarse granularity rather than geographical coordinates, which limits the practicality of the prediction. The proposed method does not reduce granularity, and predicted trajectories have the same character as historical trajectories. Our evaluation shows results on both labeled and geographical trajectories with a prediction error reduced by 10-35% compared to the baselines. This improvement has the potential to enable precise location services, raising usefulness to users to unprecedented levels.

6.2 Future research

In this dissertation, we have analysed user data individually without considering the impact of the users on each other. In fact, for each user, one model is trained based on the data from that user. For future work, one can leverage the relationship between the mobility patterns of the users to improve the results of the analysis. For example, having information about a user's friend location can improve the prediction accuracy.

Using other contextual factors in addition to the location also enhances the mining of a user's mobility pattern. By leveraging other types of data, performance can be improved. For example, considering smartphone logs can help us to infer the user's location. The user may use specific apps at specific locations. The call log is another contextual factor that, if analysed in a correct way, can improve the performance of the analysis. In the Device Analyser dataset, using context values such as smartphone logs, sensor data, and a user's activities could make the trajectory prediction and transition time detection more accurate.

Future work would involve improving and specialising the techniques proposed in this thesis.

- Regarding temporal data, the next step after temporal segmentation can be temporal clustering, where similar segments are grouped together. From a human mobility point of view, the segments in one cluster denote the same activity such as working at the office. For future work, it would also be possible to investigate how to utilise the proposed algorithm in an incremental mode for online processing.

- *Shrink* is a generic way for preserving distance in graph summarisation. However, Shrink's performance can be improved by focusing on the summarisation of movement graphs. One of the important issues is finding the close and similar nodes in the movement graph for merging. In practice, these nodes denote the same location (e.g. CIDs in a campus) and merging them results in a reasonable coarse graph. *Shrink* node selection criteria concentrate on preserving the distances while if another approach identifies which nodes should be merged, the performance improves for mining human mobility patterns.

- To better perform in terms of trajectory prediction, a more complex model can be deployed. The model should be able to compare trajectories by extracting features. Similar to our method, the predicted trajectory should be similar to the trajectory with a similar morning part. Another way of improving performance is to analyse the user's behaviour before applying the prediction method. In the proposed method, we assume that the morning trajectory is an informative clue in the prediction of an afternoon trajectory. This assumption is generally correct. However, for some users, the trajectories from previous days may provide a better clue. Future work would involve clustering the users and training models for each cluster.

6.3 Outlook

New advances in technology and mobile devices have generated large amounts of pervasive signals in the field of human mobility. Processing this data enables us to address different questions about human mobility that can benefit numerous areas including urban computing, social science, traffic management, and controlling the spread of biological and mobile viruses. The pervasive signals for mining human mobility can be captured from different channels and hence have different characteristics.

In this dissertation, we have attempted to understand and predict human mobility by mining different types of pervasive signals. We also proposed a solution to store and analyse data when resource are limited. We hope that the new approaches proposed in this thesis will be used by researchers in various academic disciplines.

Bibliography

I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. *A hub-based labeling algorithm for shortest paths in road networks*, pages 230–241. Springer, 2011. ISBN 3642206611.

R. P. Adams and D. J. MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

C. C. Aggarwal and H. Wang. A survey of clustering algorithms for graph data. In *Managing and mining graph data*, pages 275–301. Springer, 2010.

A. V. Aho, M. R. Garey, and J. D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.

T. Akiba, Y. Iwata, and Y. Yoshida. Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 349–360. ACM, 2013.

T. Akiba, T. Hayashi, N. Nori, Y. Iwata, and Y. Yoshida. Efficient top-k shortest-path distance queries on large networks by pruned landmark labeling. In *AAAI*, pages 2–8, 2015.

I. Althfer, G. Das, D. Dobkin, D. Joseph, and J. Soares. On sparse spanners of weighted graphs. *Discrete Computational Geometry*, 9(1):81–100, 1993. ISSN 0179-5376.

T. Anagnostopoulos, C. Anagnostopoulos, and S. Hadjiefthymiades. Mobility prediction based on machine learning. In *12th IEEE International Conference on Mobile Data Management (MDM)*, volume 2, pages 27–30. IEEE, 2011.

A. Asahara, K. Maruyama, A. Sato, and K. Seto. Pedestrian-movement prediction based on mixed markov-chain model. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 25–33. ACM, 2011.

D. Ashbrook and T. Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003. ISSN 1617-4917. doi: 10.1007/s00779-003-0240-0. URL http://dx.doi.org/10.1007/s00779-003-0240-0.

O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas. Window size impact in human activity recognition. *Sensors*, 14(4):6474–6499, 2014. URL http://www.mdpi.com/1424-8220/14/4/6474/pdf.

N. Banovic, T. Buzali, F. Chevalier, J. Mankoff, and A. K. Dey. Modeling and understanding human routine behavior. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 248–260. ACM, 2016.

O. O. Barzaiq and S. W. Loke. Using self-histories to predict store visits in indoor retail environments for mobile advertising: A ranked-based technique. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 1698–1705, Aug 2015. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.309.

O. O. Barzaiq, S. W. Loke, and H. Lu. On trajectory prediction in indoor retail environments for mobile advertising using selected self-histories. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 304–307, Aug 2015. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.64.

F. Bastani, Y. Huang, X. Xie, and J. W. Powell. A greener transportation mode: flexible routes discovery from gps trajectory data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 405–408. ACM, 2011.

R. A. Baxter and J. J. Oliver. The kindest cut: minimum message length segmentation. In *Algorithmic Learning Theory*, pages 83–90. Springer, 1996.

M. A. Bayir, M. Demirbas, and N. Eagle. Mobility profiler: A framework for discovering mobility profiles of cell phone users. *Pervasive and Mobile Computing*, 6(4):435–454, 2010.

R. Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, 1961.

A. Bhattacharya and S. K. Das. Lezi-update: An information-theoretic approach to track mobile users in pcs networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 1–12. ACM, 1999.

A. Bhattacharya and S. K. Das. Lezi-update: An information-theoretic framework for personal mobility tracking in pcs networks. *Wireless Networks*, 8(2/3):121–135, 2002.

P. Boldi, M. Rosa, M. Santini, and S. Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World Wide Web*, pages 587–596. ACM, 2011.

J. V. Braun, R. Braun, and H.-G. Müller. Multiple changepoint fitting via quasilikelihood, with application to dna sequence segmentation. *Biometrika*, 87(2):301–314, 2000.

N. R. Brisaboa, S. Ladra, and G. Navarro. Compact representation of web graphs with extended functionality. *Information Systems*, 39:152–174, 2014. ISSN 0306-4379. doi: http://dx.doi.org/10.1016/j.is.2013.08.003. URL http://www.sciencedirect.com/science/article/pii/S0306437913001051.

P. Buneman, M. Grohe, and C. Koch. Path queries on compressed xml. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 141–152. VLDB Endowment, 2003.

Š. Čebirić, F. Goasdoué, and I. Manolescu. Query-oriented summarization of rdf graphs. *Proceedings of the VLDB Endowment*, 8(12):2012–2015, 2015.

J. Chan, W. Liu, A. Kan, C. Leckie, J. Bailey, and K. Ramamohanarao. Discovering latent blockmodels in sparse and noisy graphs using non-negative matrix factorisation. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 811–816. ACM, 2013.

L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502. ACM, 2005.

L. Chen, M. Lv, and G. Chen. A system for destination and future route prediction based on trajectory mining. *Pervasive and Mobile Computing*, 6(6):657–676, 2010.

M. Chen, Y. Liu, and X. Yu. Nlpmm: A next location predictor with markov modeling. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 186–197. Springer, 2014.

Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. Tung. Spade: On shape-based pattern detection in streaming time series. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 786–795. IEEE, 2007.

Y.-C. Chen, J. Kurose, and D. Towsley. A mixed queueing network model of mobility in a campus wireless network. In *INFOCOM, 2012 Proceedings IEEE*, pages 2656–2660. IEEE, 2012.

Z. Chen, H. T. Shen, and X. Zhou. Discovering popular routes from trajectories. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 900–911. IEEE, 2011.

C. Cheng, R. Jain, and E. van den Berg. Location prediction algorithms for mobile wireless systems. In *Wireless internet handbook*, pages 245–263. CRC Press, Inc., 2003.

W. Cheng, X. Zhang, F. Pan, and W. Wang. Hicc: an entropy splitting-based framework for hierarchical co-clustering. *Knowledge and Information Systems*, pages 1–25, 2015. ISSN 0219-1377. doi: 10.1007/s10115-015-0823-x.

F. Chierichetti, R. Kumar, S. Lattanzi, M. Mitzenmacher, A. Panconesi, and P. Raghavan. On compressing social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 219–228. ACM, 2009.

E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1082–1090, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020579. URL http://doi.acm.org/10.1145/2020408.2020579.

F.-l. Chung, T.-C. Fu, R. Luk, and V. Ng. Evolutionary time series segmentation for stock data mining. In *Proceedings. 2002 IEEE International Conference on Data Mining (ICDM)*, pages 83–90. IEEE, 2002.

V. Colizza, A. Barrat, M. Barthelemy, A.-J. Valleron, and A. Vespignani. Modeling the worldwide spread of pandemic influenza: baseline case and containment interventions. *PLoS medicine*, 4(1):e13, 2007.

S. K. Das and S. K. Sen. Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment. *The Computer Journal*, 42(6):473–486, 1999.

S. K. Das, D. J. Cook, A. Battacharya, E. O. Heierman, and T.-Y. Lin. The role of prediction algorithms in the mavhome smart home architecture. *IEEE Wireless Communications*, 9 (6):77–84, 2002.

M. De Domenico, A. Lima, and M. Musolesi. Interdependence and predictability of human mobility and social interactions. *Pervasive and Mobile Computing*, 9(6):798–807, 2013.

T. M. T. Do and D. Gatica-Perez. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*, 12:79–91, 2014. ISSN 1574-1192.

T. M. T. Do, O. Dousse, M. Miettinen, and D. Gatica-Perez. A probabilistic kernel method for human mobility prediction with smartphones. *Pervasive and Mobile Computing*, 20: 13–28, 2015. ISSN 1574-1192.

N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268, 2006. ISSN 1617-4909.

N. Eagle and A. S. Pentland. Eigenbehaviors: Identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066, 2009.

W. Fan, J. Li, X. Wang, and Y. Wu. Query preserving graph compression. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 157–168, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1247-9.

K. Farrahi and D. Gatica-Perez. What did you do today?: discovering daily routines from large-scale mobile data, 2008.

T. Feder and R. Motwani. Clique partitions, graph compression and speeding-up algorithms. *Journal of Computer and System Sciences*, 51(2):261–272, 1995.

P.-O. Fjällström. *Algorithms for graph partitioning: A survey*, volume 3. Linköping University Electronic Press Linköping, 1998.

A. Foss and O. R. Zaïane. A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets. In *IEEE International Conference on Data Mining*, pages 179–186. IEEE, 2002.

C. Fraley and A. E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588, 1998.

T.-c. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011. ISSN 0952-1976.

Y. Fu, H. Xiong, Y. Ge, Z. Yao, Y. Zheng, and Z.-H. Zhou. Exploiting geographic dependencies for real estate appraisal: a mutual perspective of ranking and clustering. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1047–1056. ACM, 2014.

S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez. Next place prediction using mobility markov chains. In *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*, page 3. ACM, 2012.

J. Gao, R. Jin, J. Zhou, J. X. Yu, X. Jiang, and T. Wang. Relational approach for shortest path discovery over large graphs. *Proceedings of the VLDB Endowment*, 5(4):358–369, 2011.

A. Gensler, T. Gruber, and B. Sick. Blazing fast time series segmentation based on update techniques for polynomial approximations. In *IEEE 13th International Conference on Data Mining Workshops (ICDMW)*, pages 1002–1011, Dec 2013. doi: 10.1109/ICDMW.2013.90.

F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 330–339. ACM, 2007. ISBN 1595936092.

F. Giannotti, M. Nanni, D. Pedreschi, F. Pinelli, C. Renso, S. Rinzivillo, and R. Trasarti. Unveiling the complexity of human mobility by querying and mining massive trajectory data. *The VLDB JournalThe International Journal on Very Large Data Bases*, 20(5): 695–719, 2011.

G. Gidófalvi and F. Dong. When and where next: Individual mobility prediction. In *Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, MobiGIS '12, pages 57–64, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1699-6. doi: 10.1145/2442810.2442821. URL http://doi.acm.org/10.1145/2442810.2442821.

A. Gionis and H. Mannila. Segmentation algorithms for time series and sequence data. In *Tutorial at 5th SIAM international conference on data mining*, 2005.

A. Gionis, H. Mannila, and E. Terzi. Clustered segmentations. In *Workshop on mining temporal and sequential data, 10th ACM SIGKDD international conference on knowledge discovery and data mining (KDD04)*. Citeseer, 2004.

M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

A. Gubichev, S. Bedathur, S. Seufert, and G. Weikum. Fast and accurate estimation of shortest paths in large graphs. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 499–508. ACM, 2010. ISBN 1450300995.

H. Guo, X. Liu, and L. Song. Dynamic programming approach for segmentation of multivariate time series. *Stochastic environmental research and risk assessment*, 29(1):265–273, 2015.

J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, volume 29, pages 1–12. ACM, 2000.

J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224, 2001.

M. H. Hansen and B. Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

D. Hennessey, D. Brooks, A. Fridman, and D. Breen. A simplification algorithm for visualizing the structure of complex graphs. In *Information Visualisation, 2008. IV'08. 12th International Conference*, pages 616–625. IEEE, 2008.

H. Hiisila. *Segmentation of time series and sequences using basis representations.* Thesis, 2007.

J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, and H. T. T. Toivonen. Time series segmentation for context recognition in mobile devices. In *ICDM 2001, Proceedings IEEE International Conference on Data Mining*, pages 203–210, 2001. doi: 10.1109/ICDM.2001.989520.

J. Hong, K. Park, Y. Han, M. K. Rasel, D. Vonvou, and Y.-K. Lee. Disk-based shortest path discovery using distance index over large dynamic graphs. *Information Sciences*, 382383: 201–215, 2017. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2016.12.013. URL http://www.sciencedirect.com/science/article/pii/S0020025516320308.

W.-j. Hsu, K. Merchant, H.-w. Shu, C.-h. Hsu, and A. Helmy. Weighted waypoint mobility model and its impact on ad hoc networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(1):59–63, 2005.

T. Huynh, M. Fritz, and B. Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, pages 10–19, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-136-1. doi: 10.1145/1409635.1409638. URL http://doi.acm.org/10.1145/1409635.1409638.

B. Jackson, J. D. Scargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and T. T. Tsai. An algorithm for optimal partitioning of data on an interval. *Signal Processing Letters, IEEE*, 12(2):105–108, 2005.

E. T. Jaynes. *Probability theory: The logic of science.* Cambridge university press, 2003.

H. Jeung, Q. Liu, H. T. Shen, and X. Zhou. A hybrid prediction model for moving objects. In *ICDE 2008, IEEE 24th International Conference on Data Engineering*, pages 70–79. Ieee, 2008.

H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen. Path prediction and predictive range querying in road network databases. *The VLDB Journal*, 19(4):585–602, 2010. ISSN 1066-8888.

D. Karamshuk, A. Noulas, S. Scellato, V. Nicosia, and C. Mascolo. Geo-spotting: mining online location-based services for optimal retail store placement. In *Proceedings of the 19th*

ACM SIGKDD international conference on Knowledge discovery and data mining, pages 793–801. ACM, 2013.

Y. Kawahara and M. Sugiyama. Sequential change-point detection based on direct density-ratio estimation. *Statistical Analysis and Data Mining*, 5(2):114–127, 2012.

A. Kehagias, E. Nidelkou, and V. Petridis. A dynamic programming segmentation procedure for hydrological and environmental time series. *Stochastic Environmental Research and Risk Assessment*, 20(1-2):77–94, 2006.

E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and knowledge discovery*, 7(4):349–371, 2003.

E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Sigmod Record*, 30(2):151–162, 2001a.

E. Keogh, S. Chu, D. Hart, and M. Pazzani. An online algorithm for segmenting time series. In *Proceedings IEEE International Conference on Data Mining (ICDM)*, pages 289–296, 2001b. doi: 10.1109/ICDM.2001.989531.

E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. In *Data mining in time series databases*, pages 1–21. World Scientific, 2004.

E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289. ACM, 2000.

S.-W. Kim, J.-I. Won, J.-D. Kim, M. Shin, J. Lee, and H. Kim. Path prediction of moving objects on road networks through analyzing past trajectories. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 379–389. Springer, 2007.

R. Kitamura, C. Chen, R. M. Pendyala, and R. Narayanan. Micro-simulation of daily activity-travel patterns for travel demand forecasting. *Transportation*, 27(1):25–51, 2000.

N. Kiukkonen, J. Blom, O. Dousse, D. Gatica-Perez, and J. Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. *Proc. ICPS, Berlin*, 2010.

J. Kleinberg. Computing: The wireless epidemic. *Nature*, 449(7160):287–288, 2007.

J. Kleinberg, C. Papadimitriou, and P. Raghavan. Segmentation problems. *Journal of the ACM (JACM)*, 51(2):263–280, 2004.

J. Krumm and A. B. Brush. Learning time-based presence probabilities. In *International Conference on Pervasive Computing*, pages 79–96. Springer, 2011.

J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. *UbiComp 2006: Ubiquitous Computing*, pages 243–260, 2006.

N. Kumar, V. N. Lolla, E. Keogh, S. Lonardi, C. A. Ratanamahatana, and L. Wei. Time-series bitmaps: a practical visualization tool for working with large time series databases. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 531–535. SIAM, 2005.

A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.

O. D. Lara and M. A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys and Tutorials*, 15(3):1192–1209, 2013.

J.-K. Lee and J. C. Hou. Modeling steady-state and transient behaviors of user mobility: formulation, analysis, and application. In *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 85–96. ACM, 2006.

K. LeFevre and E. Terzi. Grass: Graph structure summarization. In *SDM*, pages 454–465. SIAM, 2010.

P.-R. Lei, T.-J. Shen, W.-C. Peng, and J. Su. Exploring spatial-temporal trajectory model for location prediction. In *12th IEEE International Conference on Mobile Data Management (MDM)*, volume 1, pages 58–67. IEEE, 2011.

J. Leskovec. Stanford network analysis project. https://snap.stanford.edu/data/.

J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006.

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*, 6(1):111–121, 2012.

D. Lian, Y. Ge, F. Zhang, N. J. Yuan, X. Xie, T. Zhou, and Y. Rui. Content-aware collaborative filtering for location recommendation based on human mobility data. In *IEEE International Conference on Data Mining (ICDM)*, pages 261–270. IEEE, 2015.

T. W. Liao. Clustering of time series dataa survey. *Pattern recognition*, 38(11):1857–1874, 2005.

R. LiKamWa, Y. Liu, N. D. Lane, and L. Zhong. Moodscope: Building a mood sensor from smartphone usage patterns. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 389–402. ACM, 2013.

G. Liu and G. Maguire Jr. A class of mobile motion prediction algorithms for wireless mobile computing and communication. *Mobile Networks and Applications*, 1(2):113–121, 1996.

M. Lohrey, S. Maneth, and R. Mennicke. Xml tree structure compression using repair. *Information Systems*, 38(8):1150–1167, 2013. ISSN 0306-4379. doi: http://dx.doi.org/10.1016/j.is.2013.06.006. URL http://www.sciencedirect.com/science/article/pii/S0306437913000963.

J. Manweiler, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi. Predicting length of stay at wifi hotspots. In *INFOCOM, 2013 Proceedings IEEE*, pages 3102–3110. IEEE, 2013.

H. Maserrat and J. Pei. Neighbor query friendly compression of social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 533–542. ACM, 2010.

W. Mathew, R. Raposo, and B. Martins. Predicting future locations with hidden markov models, 2012.

D. S. Matteson and N. A. James. A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, 109(505): 334–345, 2014. ISSN 0162-1459.

J. McInerney, S. Stein, A. Rogers, and N. R. Jennings. Breaking the habit: Measuring and predicting departures from routine in individual human mobility. *Pervasive and Mobile Computing*, 9(6):808–822, 2013. ISSN 1574-1192. doi: http://dx.doi.org/10.1016/j.pmcj.2013.07.016. URL http://www.sciencedirect.com/science/article/pii/S1574119213000989.

J. Meng, Y. Hu, G. Shou, Z. Guo, and J. Huang. Research on wifi user dwell time distribution. In *IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and IEEE 12th Intl Conf on Autonomic and Trusted Computing and IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 1120–1126. IEEE, 2015.

A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–646. ACM, 2009.

T. Mori, Y. Nejigane, M. Shimosaka, Y. Segawa, T. Harada, and T. Sato. Online recognition and segmentation for time-series motion with hmm and conceptual relation of actions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 3864–3870. IEEE, 2005.

M. Morzy. Mining frequent trajectories of moving objects for location prediction. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 667–680. Springer, 2007.

D. M. Moyles and G. L. Thompson. An algorithm for finding a minimum equivalent graph of a digraph. *Journal of the ACM (JACM)*, 16(3):455–460, 1969.

B. Naveh. Jgrapht. *http://jgrapht.sourceforge.net*.

S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 419–432. ACM, 2008.

A. J. Nicholson and B. D. Noble. Breadcrumbs: forecasting mobile connectivity. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 46–57. ACM, 2008.

A. Noulas, S. Scellato, N. Lathia, and C. Mascolo. Mining user mobility features for next place prediction in location-based services. In *IEEE 12th international conference on Data mining (ICDM)*, pages 1038–1043. IEEE, 2012.

V. Panagiotou. *Blind segmentation of timeseries: A two-level approach.* Thesis, 2015.

T. Pavlidis and S. L. Horowitz. Segmentation of plane curves. *IEEE transactions on Computers*, (8):860–870, 1974.

J. Peajcariaac and Y. Tong. *Convex Functions, Partial Orderings, and Statistical Applications.* Elsevier Science, 1992. ISBN 9780080925226. URL https://books.google.com.au/books?id=rCAOFpic7AkC.

J. Petzold, F. Bagci, W. Trumler, and T. Ungerer. Comparison of different methods for next location prediction. In *European Conference on Parallel Processing*, pages 909–918. Springer, 2006.

H. Pham, C. Shahabi, and Y. Liu. Ebm: an entropy-based model to infer social strength from spatiotemporal data. In *Proceedings of the 2013 international conference on Management of data*, pages 265–276. ACM. ISBN 1450320376.

D. Rafiei. Effectively visualizing large networks through sampling. In *Visualization, 2005. VIS 05. IEEE*, pages 375–382. IEEE, 2005.

S. Raghavan and H. Garcia-Molina. Representing web graphs. In *Proceedings. 19th International Conference on Data Engineering*, pages 405–416. IEEE, 2003.

E. G. Ravenstein. The laws of migration. *Journal of the statistical society of London*, 48(2): 167–235, 1885.

A. I. J. T. Ribeiro, T. H. Silva, F. Duarte-Figueiredo, and A. A. Loureiro. Studying traffic conditions by analyzing foursquare and instagram data. In *Proceedings of the 11th ACM symposium on Performance evaluation of wireless ad hoc, sensor, & ubiquitous networks*, pages 17–24. ACM, 2014.

V. Roth, T. Lange, M. Braun, and J. Buhmann. A resampling approach to cluster validation. In *Compstat*, pages 123–128. Springer, 2002.

N. Roy, T. Gu, and S. K. Das. Supporting pervasive computing applications with active context fusion and semantic context delivery. *Pervasive and Mobile Computing*, 6(1):21–42, 2010.

N. Ruan, R. Jin, and Y. Huang. Distance preserving graph simplification. In *IEEE 11th International Conference on Data Mining (ICDM)*, pages 1200–1205. IEEE, 2011.

A. Sadilek and J. Krumm. Far out: Predicting long-term human mobility. In *AAAI*, 2012.

A. Sadri. Mining changes in mobility patterns from smartphone data. In *PerCom Workshops*, pages 1–3. IEEE, 2016.

A. Sadri, Y. Ren, and F. D. Salim. Full trajectory prediction: What will you do the rest of the day? In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. ACM, 2017.

S. Salvador and P. Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 576–584. IEEE. ISBN 076952236X.

A. Sanfeliu and K.-S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (3):353–362, 1983.

S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell. *NextPlace: A Spatio-temporal Prediction Framework for Pervasive Systems*, pages 152–169. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-21726-5. doi: 10.1007/978-3-642-21726-5_10. URL http://dx.doi.org/10.1007/978-3-642-21726-5_10.

R. Schenkel, A. Theobald, and G. Weikum. *HOPI: An efficient connection index for complex XML document collections*, pages 237–255. Springer, 2004. ISBN 3540212000.

J. Scott, A. Bernheim Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar. Preheat: controlling home heating using occupancy prediction. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 281–290. ACM, 2011.

R. Scott Harris, D. R. Hess, and J. G. Venegas. An objective analysis of the pressure-volume curve in the acute respiratory distress syndrome. *American journal of respiratory and critical care medicine*, 161(2):432–439, 2000. ISSN 1535-4970.

B.-S. Seah, S. S. Bhowmick, C. F. Dewey, and H. Yu. Fuse: a profit maximization approach for functional summarization of biological networks. *BMC bioinformatics*, 13(3):1, 2012.

C. E. Shannon and W. Weaver. *The mathematical theory of communication.* University of Illinois press, 2015. ISBN 025209803X.

P. Smyth. Clustering using monte carlo cross-validation. In *KDD*, pages 126–133, 1996.

P. Smyth. Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and computing*, 10(1):63–72, 2000.

C. Sommer. Shortest-path queries in static networks. *ACM Computing Surveys (CSUR)*, 46 (4):45, 2014. ISSN 0360-0300.

C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

L. Song, D. Kotz, R. Jain, and X. He. Evaluating next-cell predictors with extensive wi-fi mobility data. *IEEE Transactions on Mobile Computing*, 5(12):1633–1649, 2006.

X. Song, Q. Zhang, Y. Sekimoto, T. Horanont, S. Ueyama, and R. Shibasaki. Modeling and probabilistic reasoning of population evacuation during large-scale disaster. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239. ACM, 2013.

X. Su, H. Tong, and P. Ji. Activity recognition with smartphone sensors. *Tsinghua Science and Technology*, 19(3):235–249, 2014.

F.-T. Sun, Y.-T. Yeh, H.-T. Cheng, C.-C. Kuo, and M. Griss. Nonparametric discovery of human routines from sensor data. In *(PerCom), IEEE International Conference on Pervasive Computing and Communications*, pages 11–19. IEEE, 2014.

S. Thajchayapong and J. M. Peha. Mobility patterns in microcellular wireless networks. volume 5, pages 52–63. IEEE, 2006.

Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 567–580. ACM, 2008. ISBN 160558102X.

H. Toivonen, F. Zhou, A. Hartikainen, and A. Hinkka. Compression of weighted graphs. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 965–973. ACM, 2011.

K. Toohey and M. Duckham. Trajectory similarity measures. *SIGSPATIAL Special*, 7(1): 43–50, May 2015. ISSN 1946-7729. doi: 10.1145/2782759.2782767. URL http://doi.acm.org/10.1145/2782759.2782767.

L. H. Tran, M. Catasta, L. K. McDowell, and K. Aberer. Next place prediction using mobile data. In *Proceedings of the Mobile Data Challenge Workshop (MDC 2012)*, number EPFL-CONF-182131, 2012.

S. J. van Schaik and O. de Moor. A memory efficient reachability data structure through bit vector compression. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 913–924. ACM, 2011.

K. T. Vasko and H. T. Toivonen. Estimating the number of segments in time series data using permutation tests. In *IEEE International Conference on Data Mining (ICDM)*, pages 466–473. IEEE, 2002.

D. Wackerly, W. Mendenhall, and R. Scheaffer. *Mathematical statistics with applications*. Cengage Learning, 2007.

D. T. Wagner, A. Rice, and A. R. Beresford. Device analyzer: Large-scale mobile data collection. *ACM SIGMETRICS Performance Evaluation Review*, 41(4):53–56, 2014. ISSN 0163-5999.

D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1100–1108. ACM, 2011.

H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu. Path problems in temporal graphs. *Proc. VLDB Endow.*, 7(9):721–732, May 2014. ISSN 2150-8097. URL http://dl.acm.org/citation.cfm?id=2732939.2732945.

X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning*, pages 1033–1040. ACM, 2006.

L. Yao, Q. Z. Sheng, W. Ruan, X. Li, S. Wang, and Z. Yang. Unobtrusive posture recognition via online learning of multi-dimensional rfid received signal strength. In *International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2015.

F. Yu and V. Leung. Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks. *Computer Networks*, 38(5):577–589, 2002.

H. Yu, C. Li, and J. Dauwels. Network inference and change point detection for piecewise-stationary time series. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4498–4502. IEEE, 2014.

J. Yuan, Y. Zheng, and X. Xie. Discovering regions of different functions in a city using human mobility and pois. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 186–194. ACM, 2012.

J. Yuan, Y. Zheng, X. Xie, and G. Sun. T-drive: enhancing driving directions with taxi drivers' intelligence. volume 25, pages 220–232, 2013.

M. Zhang and A. A. Sawchuk. Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *ACM International Conference on Ubiquitous Computing (Ubicomp) Workshop on Situation, Activity and Goal Awareness (SAGAware)*, Pittsburgh, Pennsylvania, USA, September 2012.

N. Zhao, W. Huang, G. Song, and K. Xie. Discrete trajectory prediction on mobile data. In *Asia-Pacific Web Conference*, pages 77–88. Springer, 2011.

Y. Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

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*, WWW '09, pages 791–800, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526816. URL http://doi.acm.org/10.1145/1526709.1526816.

Y. Zheng, X. Xie, and W.-Y. Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.

Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3): 38, 2014.

F. Zhou, S. Malher, and H. Toivonen. Network simplification with minimal loss of connectivity. In *IEEE 10th International Conference on Data Mining (ICDM)*, pages 659–668. IEEE, 2010.

A. D. Zhu, X. Xiao, S. Wang, and W. Lin. Efficient single-source shortest path and distance queries on large graphs. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 998–1006. ACM, 2013.

C. J. Zhu, K.-Y. Lam, and S. Han. Approximate path searching for supporting shortest path queries on road networks. *Information Sciences*, 325:409–428, 2015. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2015.06.045. URL http://www.sciencedirect.com/science/article/pii/S0020025515004776.

J. Zhuang, T. Mei, S. C. Hoi, Y.-Q. Xu, and S. Li. When recommendation meets mobile: contextual and personalized recommendation on the go. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 153–162. ACM, 2011.