On the Effectiveness of Removing Location Information from Trajectory Data for Preserving Location Privacy

by

Amina Hossain

Submitted in the partial fulfillment of
the requirements for the degree of

Master of Philosophy - Engineering

Department of Computing and Information Systems
The University of Melbourne

December 5, 2017

Abstract

Thesis Title: On the Effectiveness of Removing Location Information from Trajectory Data for Preserving Location Privacy

Advisors: Prof. Lars Kulik, Assoc. Prof. Egemen Tanin

The ubiquity of GPS enabled smartphones with Internet connectivity has resulted in the widespread development of location-based services (LBSs). People use these services to obtain useful advises for their daily activities. For example, a user can open a navigation app to find a route that results in the shortest driving time from the current location to a destination. Nevertheless, people have to reveal location information to the LBS providers to leverage such services. Location information is sensitive since it can reveal habits about an individual. LBS providers are aware of this and take measures to protect user privacy. One well established and simple approach is to remove GPS data from user data working with the assumption that it will lead to a high degree of privacy. In this thesis, we challenge this notion of removing location information while retaining other features would lead to a high degree of location privacy. We find that it is possible to reconstruct the original routes by analyzing just the turn instructions provided to a user by a navigation service. We evaluated our approach using both synthetic and real road network data and demonstrate the effectiveness of this new attack in a range of realistic scenarios.

Declaration

This is to certify that

- the thesis comprises only my original work,
- due acknowledgment has been made in the text to all other material used,
- the thesis is less than 50,000 words in length, exclusive of tables, maps, bibliographies, appendices and footnotes.

.....

Signature

Date: .....

Dedicated to my Parents

Acknowledgment

First, I would like to thank my supervisors, Prof. Lars Kulik and Assoc. Prof. Egemen Tanin, for their continuous guidance and support, without which the work presented in this thesis would not have been accomplished. Influence and motivation provided by them throughout the candidature helped me to improve my research skills significantly.

I am grateful to Assoc. Prof. Harald sondergaard for being in my MPhil advisory committee chair and spend his valuable time to review my works.

Next, my spatial thanks go to Anthony Quattrone who was a co-author in my research paper, help me to produce experiments' results for real road data related to this research. His expertise and knowledge helped me to look at a problem deeply and improve the presentation of my work.

I do not have words to express my gratitude to my parents for their unconditional love, affection and support throughout my life. It would not have been possible to reach at this stage without their sacrifice and contribution.

My deepest thanks to my elder brother and sister, who always inspire and support me to continue the success in my life.

I express my gratitude to my husband for his great support, care and guidance during my hardship and bad times of my MPhil.

Next, my sincere gratitude goes to Elham Naghizade and Maryam Fanaeepour, for their great assistance during learning period of the programming language Python first time and also appreciate their warm friendship.

My special thanks goes to Kate Hale, Oleksii Verdernikov, Ishita Akhter and Nick May for their helping hand to review my thesis's chapters.

I would like to thank to the fellow research students, Kushani Anuradha Perera, Tanussri Bhattachariya, Hengfeng Li, Lida Rashidi, Hairuo Xie for their warm friendship, which made my stay in the university an unforgettable experience.

I take this opportunity to thank the University of Melbourne for granting me an IPRS and APA (Int) scholarships, which helped me to attend one of the most sought after Universities in the world and allowing me to stay focused on this research without facing financial hardships. My sincere gratitude goes to the staff of the Department of Computing and Information Systems for providing a rich and friendly academic environment to conduct this research work successfully.

Finally, I would like to express my gratitude to Almighty for His mercy that allows me to successfully accomplish the Mphil.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Smartphones are capable of providing accurate positioning estimates. Combined with the ability to connect to the Internet at high speeds, this has led to the widespread development of location-based services (LBSs). LBSs make use of the location information and consider finding a service that is more applicable to location. Examples of LBSs include local business search (e.g., searching for restaurants within a user specified range distance from a user), social networking (e.g., a group of friends sharing their geo-tagged messages), automotive traffic monitoring (e.g., inferring traffic congestion from the position and speed information periodically reported from probe vehicles), and route finder applications (e.g., searching a route with the shortest driving time between two locations). In particular, LBSs provide important way-finding services for users to assist them in navigation while people are travelling. However, at the same time the location of the user being sent to LBSs opens up the possibility of exposing them to greater privacy risks. Such frequent and continuous access to these services further exposes and exacerbates privacy risks for the location of the user. The LSP can produce a complete profile of the user's movement with a high degree of spatial and temporal precision, possibly over an extended period of time. From this profile, the LSP may infer private and sensitive information about the user. For example, if a user is at drug rehab center then the LSP might infer that the user has had a drug addiction or if the user's location is at a stadium during office hours, the LSP could predict the user's level of sincerity for his work.

Location privacy has been defined as a spatial type of information privacy that con-

cerns with the claim of individuals to determine for themselves when, how and to what extent location information about them is communicated to others [24]. This definition clearly pinpoints the pivotal role of individuals in controlling the way their location data is being utilized. Therefore, with the popularity of LBSs, user privacy protection becomes an important research issue for future generation communication and networking. The privacy threat is exacerbated if the available information from the access of LBS is linked with other data sources; in the former example, if the LSP collaborates with the health service provider then more specific information about the user's addiction to a drug could be revealed. Knowing about a user's drug addiction can cause difficulties in a user's life.

Presently, an extensive survey [1] has been done by Microsoft for assessing the state of the art of LBSs. The study showed that more than 50% users are concerned about privacy risks that could arise from accessing LBSs though 92% users think LBSs as valuable services. The main concern of the users comes from sharing location with unspecified organizations or people. The advent of Google's Android operating system and Apple's iPhone OS allow third parties to run their applications and become LSPs for the users of mobile devices. LSPs provide APIs so small developers can act as a proxy LBS. The development of Android and iOS is not the main reason, although more people accessed these services once smartphones proliferated. Thus, the LSPs are not only limited to large corporations such as Google or Microsoft but include small developers, companies or the employers. For example, an organization may provide LBSs for its employees, that indirectly reveals that one employee is at a different company for a job interview, a fact an employee wish to not disclose at that time. In this thesis, we highlight methods that enable a user to enjoy a high quality of service without revealing private user information to an LSP.

Numerous studies have been conducted for preserving location privacy of individual. A popular privacy model to hide a user's location from LSP is to send a cloaked area [10, 18, 38, 91], which is typically a rectangle containing the user's location, instead of the user's location. The higher the spatial imprecision is, the more private is the user's location; the probability is higher that a larger rectangle area contains a diverse range of locations. Another popular way to preserve user location privacy is to hide a user's identity instead of her location and considers that the LSP might have knowledge about the user's location and then the rectangle is computed based on $K-$anonymity [34, 40]. It

ensures that the rectangle includes the locations of at least $K-1$ other users in addition to the user's location so that the user becomes K-anonymous.

A trajectory is the location variation of an object over time. A trajectory is represented as a time-ordered sequence of points, each point referring to a geographic location [87]. Geographic location refers to a specific physical point on Earth. This is more precise than "area" or "place", often defined by a set of latitude and longitude coordinates. Fundamentally, a trajectory data consists of the user's highly sensitive location information. The sensitiveness of handling trajectory data has been clearly stated in [31]. Trajectory privacy refers to preventing unauthorized parties from learning user's current or past trip or parts of it.

Large quantities of trajectory data are generated every day by smartphone users during their travels. This data could be considered as very private due to having location information of an individual user. For example, when users request location based queries while they are travelling on the road they must reveal their identities, location point or trajectory (sequence of location points) and the queries describing the required service, from which an LSP might be able to derive sensitive and private information about a user's health, habits, and preferences. Let us assume, if a user requests a query for the nearest heart centre then the user's health condition could be revealed to the LSP. The straightforward solution is to protect the user's privacy by hiding the user's identity which is inferred from the IP address or the mobile phone number. However, hiding only the identity cannot always protect a user's privacy because an LSP can have knowledge about a user's location and the user's identity could be inferred from the revealed location. If the user requests the query from her home or office then the revealed location that is the home or office address can act as an identifier of the user. The other source of knowledge for a user's location can be physical observation or mobile phone triangulation. Furthermore, there are cases where a user might request a personalized service which means the user's identity is revealed to the LSP while accessing a service. As an example, when a user's identity is revealed while requesting a query for the nearest gas station, the LSP can only return those gas stations which provide a higher discount for the user's credit card. In this scenario, the LSP might not have knowledge about the user's location; instead the location might disclose sensitive and private information about the user. In summary, the user's location can sometimes reveal the user's identity and sometimes act as a source

of the user's private and sensitive data. Thus, hiding the user's location while requesting a location-based query is essential to protect the user's privacy.

Techniques demonstrating how to preserve the trajectory privacy of LBS users has been proposed in [4, 10, 22, 34, 35, 36, 40]. Most of the trajectory privacy preserving techniques are based on the Spatial cloaking method that relies on $K-$anonymity concept and cloaking granularity, which obfuscates a user's location into a cloaked spatial area that satisfies the user's specified privacy requirements. Commonly used the trajectory privacy preserving methods including Path confusion technique, Dummy trajectories technique. The overall concept of these existing works is to preserve users' trajectory privacy by removing location information at different levels from the trajectory data that users reveal while they are using LBSs. However, in this thesis we show that even removing all the location information from the trajectory data is insufficient to protect users' privacy in a range of scenarios.

Trajectory data sets consist of not only user trajectory information but also relevant navigational advice given to users. Navigational advice refers to the set of route instructions for getting the individual's position and planning and following a trajectory. For example, people could find shortest route [20] or even simplest route [21] to their destination with the help of navigational advices from the navigation services of LBSs. This advice can include turn instructions supplied to a user for navigation purposes. We are particularly interested in the turn instructions in this thesis. Turn instructions are a primary means to guide a user in order for them to reach to their destination. In [19, 79, 84], turn instructions are defined as instructions on how to follow a route by providing task-oriented specifications of the actions to be followed to reach a destination. These instructions can easily be argued as belonging to the LSP and could be retained for a long period of time or traded.

Plenty of studies [17, 21, 32, 33, 44, 64, 65, 66, 88] have been conducted on generating route instructions. Shortest path algorithms (will discuss in Chapter 3) are widely used to generate route instructions and it addresses the issue of finding the shortest route for the user. However, the shortest route may involve a number of intersections that are difficult to navigate, because they offer more than one alternative to turn left or right. Some researches [21, 44, 66] have focused on the issue of finding simple path in terms of navigation advice complexity. Generating route instructions has also been studied from

a natural language processing point of view in [17, 64, 65]. The authors in [17] utilize hierarchical Markov Decision Processes (MDP) [5, 72] to adaptively derive instructions and [64, 65] use machine learning to derive natural language route instructions from a previously learned corpus of instruction expressions. A range of works [32, 33] have also exploited turning angles and distances to accurately predict the actual route a user has traveled along. The use of path shapes was proposed in [32] where authors create a shape represented by angles and distances and find similar shapes as derived from the road network to map match the trajectory a user travelled along. This work was extended in [33] to use only angle information to perform successful map matching. The use of turn instructions has been studied in [88] to determine the probability of a user reaching their destination. Coarse turn instructions such as (left, right, straight) can be ambiguous given the road network context. The authors suggest to use instructions such as sharp left or slightly right. In our work, we use very coarse turn instructions to demonstrate that even sharing this information can present a privacy issue as the users' actual route can be identified.

We present a new approach that uses turn instructions to find the actual route of a user. The reason for utilising the concept of turn instructions in our work is twofold: firstly, turn instructions are commonly correlated with the road network environment. Secondly, these instructions may be exchanged by the LBS with third parties without significant legal implications [25]. In this thesis, we show that by removing location information and retaining just the meta-data including turn instructions does not always preserve privacy in different realistic scenarios.

1.2 Research Question

Trajectory data is highly impressionable due to having a direct association with user location information. Therefore, many studies [2, 4, 10, 22, 34, 35, 36, 40, 50, 68] have aimed to remove the user's location information from trajectory data to obfuscate the user's identity from the LBS service provider or third party. However, trajectory data is a combination of individual location information with relevant features of that trajectory, such as turn directions that describe how to navigate the trajectory. Our goal was to develop an approach for removing location information from trajectory data and

retaking directional information so that we can reverse engineer the trajectory information from directions. Most existing trajectory privacy preserving methods work with GPS coordinates of the trajectory data, and Map-matching algorithms (for example [59, 60]) are used to convert GPS coordinates into a sequence of edges in the graph. We use the sequence of edges in a graph that represent the road network and then compute the turn instructions for every pair of edges in a graph by using turn instruction generation algorithm. We then investigate the following question:

- What is a method for finding out trajectory data from the road network using driving instructions? To answer this question, we investigate the factors involved in reconstructing routes based on turn instructions without using the location information of users.

1.3 Contributions

Currently, most navigation apps are integrated with turn instructions for providing assistance to road users during their journey. Privacy preserving techniques assist only in removing location information i.e., GPS coordinates of a user from the users' trajectory data sets while they seek advice from LBS services, in a manner that preserves users' location privacy. Nevertheless, these trajectory data sets also contain information associated with the user location trajectory, such as driving distance, speed limit, and navigation instructions for example left, straight, right turn instructions for following the trajectory. We explore associative information of trajectory data sets of a user that can be utilised by a third party to reconstruct the path. To the best of our knowledge, our work is the first to address this important issue. In this thesis, we show that our proposed method, based on turn instructions of the trajectory, can efficiently reconstruct the path from the map without any positional knowledge of the user.

This thesis offers the following major contributions:

- We show that it is possible for a trajectory data set that does not have GPS or other specific location information, to access these details from turn instructions. The proposed method exploits existing trajectory data from the fine-grained GPS coordinates of the digital map and represents these data into road segments or corresponding edges of the graph. We generate turn instructions for every pair of road

segments by using a turn instructions generation algorithm. Existing approaches rely on computing turn instructions to improve the level of accuracy. We address the user location privacy infringement that is generated from retaining the information of the trajectory data so that we use coarse turn instructions, i.e., straight, left and right, in our work. In the final step, we introduce a method for interpreting turn instructions from the trajectory data to find appropriate paths without the users' positional information and demonstrate the robustness of our proposed method.

- We show that companies and researchers should not solely rely on removing location information, as turning directions can be used to find out location information.

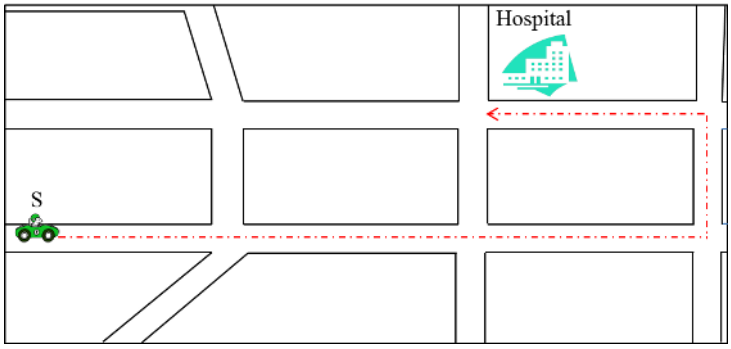

Figure 1.1: A scenario of revealing a user's actual trajectory by following turn instructions given by the user

1.4 Proposed Methodology

The main idea of our methodology is to create a setting where people go from one location to another, get that data into a form that there is no direct location information, and test whether without that information we reverse engineer a user's actual travelled path. Consider the road network in Figure 1.1. A user, Bob, wishes to go to a *Hospital* from his current position, S. He may ask the LBS to provide him with a route to the *Hospital* and follows the straight, straight, left and left turn directions, sequentially (red dashed line). Even when removing the location tracks, just by using the city map and turn instructions, we can reveal that Bob started from S and ended up at the nominated destination because this sequence is unique for this city. This simple scenario is an intuitive example of a

situation where it is feasible to get high-quality location information without the use of positional information.

In this context, we consider some real road network scenarios in the Melbourne CBD (collected from the OpenStreetMap [43]): our observation is that when there are more turn instructions, the route is more likely to be unique. Let us assume we remove the location information in Figure 1.2 and we keep the turn instruction information. A user, Alice, wishes to go to a *Burger Shop* from the corner of *Elizabeth Street* and *Lonsdale Street*. She may ask the LBS to give her the route that can follow straight, right, straight and right directions sequentially and she receives one route (red line).

Figure 1.2: Choosing turn instructions in a real road network

Alternately, let us consider the scenario in Figure 1.3. Alice decides to go to a *Burger Shop* by using fewer turn instructions from the corner of *Elizabeth Street* and *Lonsdale Street*. For this, she may ask the LBS to give her the route that can follow straight, right and straight directions consecutively and she receives two alternative routes (red and yellow lines). As we can see, if we utilise the longer turning instructions (straight, right, straight and right), we find that there is only one possible path that follows the given instructions by Alice. Looking at the shorter turning instructions, there are several possible destinations. Thus, the shorter turn instructions resultin more privacy for the user than the longer turn instructions. In this thesis, our main focus is to point out the extent to which exchanging users' trajectory information preserves privacy. We have a plan to examine how to protect the user's location privacy from the above scenario in

near future.

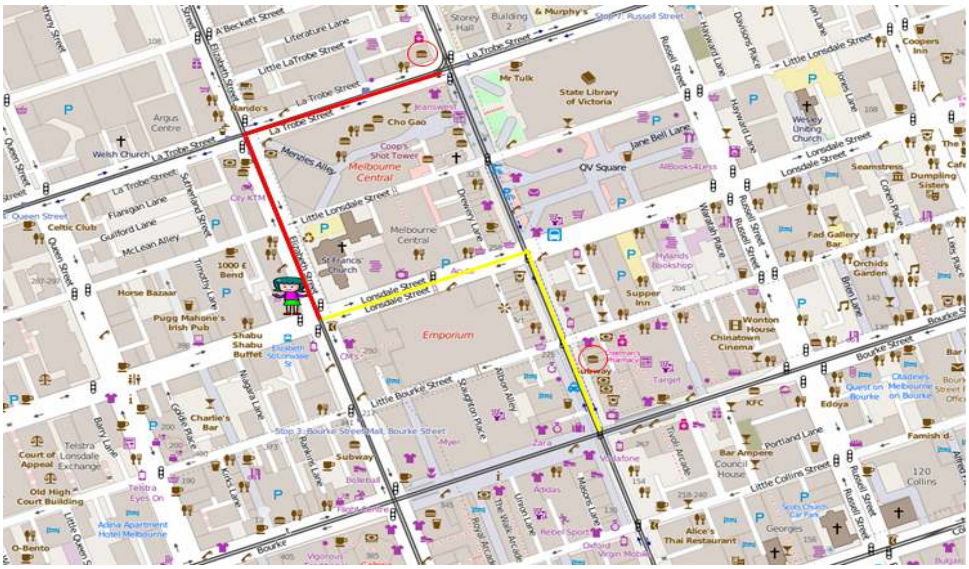

Figure 1.3: A scenario of obtaining multiple trajectories with the same turning instructions provided by Alice

We compute route instructions based on three basic turn instructions, i.e., "Left", "Straight" and "Right" for given paths. Then we check for each possible source node in a road network if the sequence of turn instructions can be traversed and leads to a final destination. Our method tries to apply consecutively each turn instruction and stops when a turn instruction cannot be applied. If all instructions be executed, the algorithm terminates, and a route is stored as a candidate trip. The number of resulting equivalent candidate routes indicates the privacy level of a user's travelled route as the user could have travelled on any one of them.

Then we evaluate the efficacy of our proposed approach by considering for a given route i.e., a set of turn instructions, how unique this route is. We assume that the larger of number of trajectories that can be mapped to an instruction sequence, the more private this instruction sequence is. This enables us to determine those instruction sequences that can be released for a given privacy level and those that cannot be revealed to third parties. We have tested our algorithm on both synthetic and real data. We have used synthetic data to test the algorithm behavior and accuracy for various trajectory parameters. We consider node degree, trip length, trip types and grid area size as parameters in our experiments. We observe that paths with a small number of route instructions are more private than paths with a higher number of route instructions. The reason is that paths

with less route instructions are more frequent in the network.

We use real data to observe the applicability of our algorithm for real road network environment. We consider different areas in Melbourne City for the evaluation of our algorithm. We evaluate the performance of the algorithm by comparing the path's length with the number of occurrences of that path can be found in the refine network. We show that our proposed approach can effectively reconstruct routes for given turn instructions and the paths with longer turn instructions are likely unique in the network. That means there is always a possibility to disclose a user's location information if the user's route instructions are exchanged with any one.

1.5 Thesis Organization

The reminder of this thesis is organized as follows:

- **Chapter 2:** We discuss details on navigation services and present an extensive literature review regarding route instruction generation in the road network context. Then we discuss the overall existing methods for preserving the privacy of users' location trajectory in location based services (LBSs) related to this thesis. The discussion includes different privacy models, techniques to compute the user's cloaked area and approaches to predict destination during the travelling period. We break down the importance of preserving users' privacy in LBSs into three main areas: data privacy, location privacy and trajectory privacy, and survey the existing privacy preserving techniques in relation to these areas.

- **Chapter 3:** We present a background on computing the shortest trajectory or path while people are travelling on a road. The shortest paths problem is one of the most important problems in road networks. The problem is especially inevitable for drive guidance systems that include navigation directions. Our work is highly related to the road network as well as navigation directions. In this chapter, we discuss existing state of the art solutions for solving shortest paths problem.

- **Chapter 4:** We propose our method in this chapter. We use the concept of turn instructions. Turn instructions refer to the actions of people as they travel. We use a turn instructions generation algorithm to compute turn instructions to reconstruct

the actual path of a user to reverse engineer location information. We introduce a turn instructions interpreting method to translate the turn instructions given by the user.

- **Chapter 5:** We describe the data we used and our experimental setup. We evaluate our proposed approach performance both for synthetic and real data sets. We experimentally show that our proposed approach achieves a higher number of paths for a given set of navigation instructions of a user while the length of navigation instructions is relatively small.

- **Chapter 6:** We conclude the thesis that user trajectory location privacy can be infringed by exchanging navigation instructions, even though the location information of the user is removed from the user's trajectory information data set. We also discuss possible future research directions.

1.6 Publication that Resulted from This Thesis

- Hossain A., Quattrone A., Tanin E. and Kulik L. (2016). On the Effectiveness of Removing Location Information from Trajectory Data for Preserving Location Privacy. In ACM SIGSPATIAL IWCTS 2016, 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science, San Francisco Bay Area, California, USA, pages 49-54.

Chapter 2

Literature Review

Location based services (LBSs) provide great benefits for users. The navigation services assist the user to determine individual's current position, determine the location points of destinations and provide directions on how to reach the destination. Navigation route descriptions include turn-by-turn instructions to reach the destination allowing people to reach their desired destination. However, these types of assistance from these navigation services could lead privacy issue of individuals' location. For example, navigation services provider could collect and collate precise details of the user's travel history for a long period of time that include location information, turn-by-turn instructions, travel distances and so forth. From the observation, the navigation service provider may determine private and sensitive information about the user. For example, if a user is at a cancer hospital then the provider might infer that the user has some form of cancer. The range of studies [2, 4, 10, 22, 34, 35, 36, 40, 50, 68] have been discussed about the problem to share the location information of individual with the service provider or a third party. The main objective of these research [2, 4, 10, 22, 34, 35, 36, 40, 50, 68] is to hide the location information of a user from a service provider or a third party as privacy preserving manner. However, the rest of the information of travel history for individual including turn instructions still remain. From this prospective, we have motivated to review the existing navigation services with route instructions and privacy preserving techniques for protecting users' location trajectory privacy in this chapter.

The remainder of this chapter is organized is as follows, we first discuss the basic concept of LBSs in Section 2.1 and then in Section 2.2, we present the importance of navigation services with route instructions for the road user of the LBSs and survey the

existing literature related this topic. Although LBSs provide valuable services for mobile users, revealing private locations to potentially untrusted LBS providers pose privacy concerns and our work is directly related with the users location privacy. We review an extensive survey of user location trajectory privacy in Section 2.3.

2.1 Location Based Services

Location-based services (LBSs) provide information and data to the user based on geographical position (i.e. spatial coordinates). LBS can be also defined as service that integrates the position of a mobile device with other information in order to provide added value to a user. A result of the convergence of information systems, communication mechanisms and positioning technologies (Figure 2.1), LBS offer a personalized approach to data access [81]. The key to LBSs is to know the location of the user, so that an appropriate service can be delivered. However, that location might not necessarily have to be related to the current position but could also be some future location of the user. The most common example of such applications include car navigation systems. These applications represent a new way of interaction between people and the virtual world. LBSs can be used in combination with messaging, for example in mobile advertising.

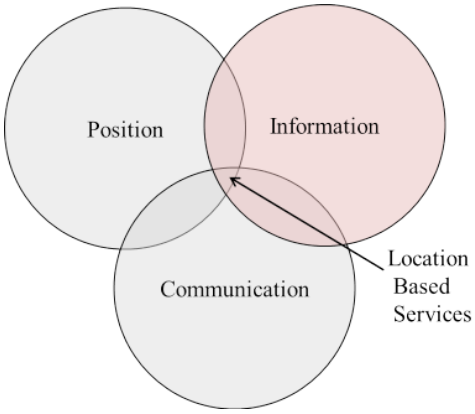

Figure 2.1: The convergence of technologies enables location based services [81]

In general, The LBSs are based on a cellular communication network and one or more positioning technologies which, combined with a geographic information system (GIS), allow the management of the collected information and its presentation to the end user. The main features of these applications are: the user positioning system, a communication network and some information. Furthermore, they are characterized by

a two-way provider-user communication, where users can interact with the provider to supply both their position and preferences regarding the requested information. The provider can thus express the most suitable answer to the users' requests.

Important classes of LBSs are those that are taking user's trajectories into account, these in composes navigation services which we will discuss in Section 2.2 but also services that depend on user real time position such as continuous nearest neighbor queries or detour queries. According to [13], there are mainly two types of LBS, namely, *snapshot* and *continuous* LBS. For *snapshot* LBS, a mobile user only needs to report its current location to a service provider once to get its desired information. On the other hand, a mobile user has to report its location to a service provider in a periodic or on-demand manner to obtain its desired *continuous* LBS.

Nevertheless, LBSs provide important services to the user, publishing personal location information to an unreliable service provider could pose potential privacy concerns. Two surveys' results regarding the concern of using LBS have been pointed out in [13] that describe 55% LBSs users show concern about their loss of location privacy and 50% of U.S. residents who have a profile on a social networking site are concerned about their privacy. Based on these results, location privacy of a user is the most significant issue for the location-dependent services. Preserving privacy in *continuous* LBSs is more challenging than *snapshot* LBSs because perpetrators could utilize the spatial and temporal correlations in the user's location samples to determine the user's actual location information. Many real life applications such as business analysis, urban planning are based on the analysis of the user location trajectories. However, publishing such location trajectories to the public or a third party for data analysis could pose serious privacy concerns. We will give an overview of the existing privacy preserving techniques in this problem in Section 2.3.

2.2 Navigation Services

Navigation service is one of the most popular service of LBSs and it must allow the user to effortlessly determine his/her own position, locate a destination, and receive guidance on how to get there [74]. The services to be developed will provide the user with guidance on the use of different transport modes and the routes reserved for them, and will also

meet user needs in areas related to travel, tourism and recreational pursuits. Based on [74], the navigation services will answer the following questions:

- Where is the user him/herself located?
- Where is the desired site or service located?
- What different modes of transport can take the user to his/her desired destination?

People are mostly dependent on the navigation applications such as car navigation is inevitable while they are on the road. The primary reason of these services popularity due to the fact that they make the essential and everyday task of travelling much easier and safer by simply providing user's current location as well as routes to the selected destinations. Navigation services would not be possible without advancements in positioning and mobile computing devices. Furthermore, technology trends today are advancing into many new and exciting areas providing opportunities that would not have been possible just a short time ago.

Navigation is often referred to the activity of accurately find out one's position and planning and following a route and it is one of the most significant aspect when users move in the road network. Human navigation research investigates the processes that take place when people orient themselves and navigate through a space [39]. Travelling through an unfamiliar road network, people may seek navigational advice from other road user as well as automated navigation systems and use various spatial, cognitive and behavioral abilities to find their paths. Current navigation services are mainly based on geometric data of the road network [75] and compute the shortest or fastest path given an origin and destination, providing a sequence of routing instructions for this path. Route instructions refer to the instructions on how to follow a route; they are task-oriented specifications of the actions to be carried out to reach a destination in [19, 79, 84]. Each single instruction is meant to guide the user from one decision point to the next. In the course of route following, a decision point is an intersection in a road network. Based on the discussion of [57], the general structure of a decision point (the physical structure), i.e., the number of brunches, angles between branches. Some examples of different general structures of a decision point are described in [57] and they are circles, T-intersections, different number of branches and highway exits.

Plenty of studies [9, 17, 21, 44, 62, 64, 65, 66, 80, 88] have been conducted on generating suitable route instructions for a road users. Shortest path algorithms (will discuss in details in Chapter 3) are widely used to generate route instructions and it addresses the issue of finding the shortest route for the individual. However, a shortest route may involve a number of intersections that are difficult to navigate, because they offer more turn left or right. For example, consider the network in Figure 2.2 (derived from [21]). This network might represent, for instance, the block structure of a road network in a city. Assuming the network is embedded in the Euclidean plane, there exist six equivalent optimal shortest routes from intersection i_1 (upper left) to intersection i_9 (lower right) i.e., $(i_1, i_2, i_3, i_6, i_9)$, $(i_1, i_2, i_5, i_6, i_9)$, $(i_1, i_2, i_5, i_8, i_9)$, $(i_1, i_4, i_5, i_6, i_9)$, $(i_1, i_4, i_5, i_8, i_9)$ and $(i_1, i_4, i_7, i_8, i_9)$, respectively. Four of these six routes i.e., $(i_1, i_2, i_5, i_6, i_9)$, $(i_1, i_2, i_5, i_8, i_9)$, $(i_1, i_4, i_5, i_6, i_9)$, $(i_1, i_4, i_5, i_8, i_9)$ seem to be required a longer instruction sequence to describe. For example, if we consider the route $(i_1, i_2, i_5, i_8, i_9)$, we might describe the route with the instruction sequence "orient yourself, go straight ahead, turn right at the first intersection and then turn left at the second intersection."

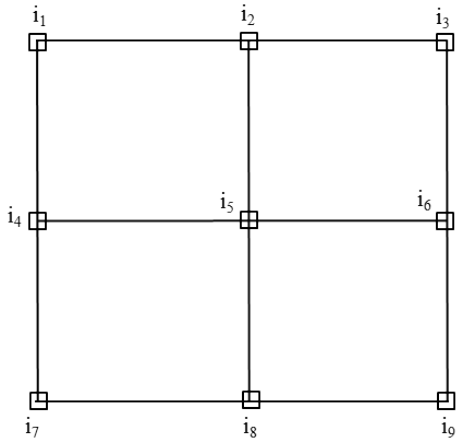

Figure 2.2: Example network with six shortest and two simplest paths [21]

A range of researches [9, 21, 44, 66, 80] have been focused on the issue of finding simple path in terms of navigation advices complexity. Take the Figure 2.2 [21] again as an example of simple path in a road network, only two of six routes ($(i_1, i_2, i_3, i_6, i_9)$, $(i_1, i_2, i_5, i_6, i_9)$, $(i_1, i_2, i_5, i_8, i_9)$, $(i_1, i_4, i_5, i_6, i_9)$, $(i_1, i_4, i_5, i_8, i_9)$ and $(i_1, i_4, i_7, i_8, i_9)$) from intersection i_1 (upper left) to intersection i_9 (lower right) seem to be optimal simple routes. Because, $(i_1, i_2, i_3, i_6, i_9)$ and $(i_1, i_4, i_7, i_8, i_9)$ avoid the more complex 4-way intersection i_5, and only these two routes require just one turn. We might describe the

route $(i_1, i_2, i_3, i_6, i_9)$ with the intersection sequence "orient yourself, go straight ahead, and turn right at the end of of the road." In contrast, the route $(i_1, i_4, i_5, i_6, i_9)$, we might describe the route with the instruction sequence "orient yourself, turn left at the first intersection, go straight ahead, and then turn right at the second intersection."

The approach in [80] can be utilized to take advantage of the type of roads (major roads versus minor roads), generating a path that prefers major routes. The time to find a route in the graph is considerable reduced, since the algorithm essentially restricts the search to major roads. The approach in [66] is a modification of the A^* shortest path algorithm [45] by considering both the total length of the route and "easy of description" of the route. Two types of intersections have used in [66]. The first category intersections according to the complexity of instructions used to describe them and accordingly adjusts the weights used in the shortest path computation to select routes through intersections that could be described utilize less complex instructions. The simplest path algorithm described in [21] aims to minimize the complexity of a route description, based on amount of information required to negotiate each decision point. Here, a shortest-path search is used in combination with a cost metric that assigns a cost value to each turn description at an intersection taking into account the number of outgoing arcs. It utilizes the notion of classification of instructions into *frames* with each frame having several *slots* for different elements of an instruction from [66], and improves on the weights used in it.

The shortest most reliable path algorithm presented in [44] attempts to minimize not the instruction complexity but the unreliability present on a path, in terms of structural complexity. It models the weights according to how many outgoing arcs are considered "instruction equivalent". This leads to their measure of unreliability as the number of instruction equivalent options minus one. The unreliability value of an entire route is then simply the sum of individual unreliability values at all intersections along the route. The unreliability value is thus zero if the instruction has no ambiguous turn instruction. Further, the authors suggest to break ties with the metric path length. Finally, their work considers a weighted sum of metric distance and unreliability value to balance reliability and route length. The landmark spider algorithm developed in [9] aims to generate a *clearest* path in terms of spatial references and landmarks used to describe the route. It exploits the same algorithm in [21] with the only difference being the weighting function for an intersection, which in this case depends on the distance and orientation of a traveller

with respect to any landmark present near the intersection.

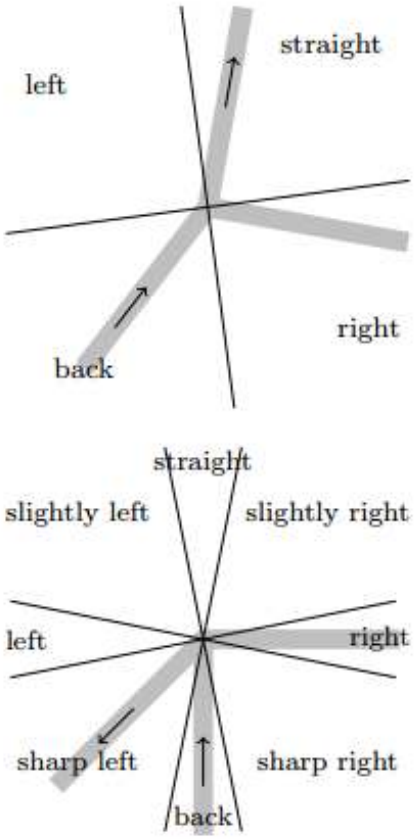

Figure 2.3: Two different approaches used to define qualitative turn descriptions for road intersections [88]

Generating route instructions has also been studied from a natural language processing point of view in [17, 64, 65]. The authors in [17] utilize hierarchical Markov Decision Processes (MDP) [5, 72] to adaptively derive instructions and [64, 65] use machine learning to derive natural language route instructions from a previously learned corpus of instruction expressions. Some works [32, 33] have also exploited turning angles and distances to accurately predict the actual route a user has travelled along. The use of path shapes was proposed in [32] where authors create a shape represented by angles and distances and find similar shapes as derived from the road network to map match the trajectory a user travelled along. This work was extended in [33] to use only angle information to perform successful map matching.

The use of turn instructions has been studied in [88] to determine the probability of a user reaching their destination. It tries to tackle the problem of efficient routing by using

cost functions that trade-off between minimizing the length of a provided path while also minimizing the number of turns on the provided path. It uses the incoming street of an intersection as the reference direction to describe the relation between this segment and the outgoing segments, and adapts the turn descriptions proposed in [55]. Naïve representation to define qualitative turn descriptions for road intersections that divides the plane into four sectors of equal size "left", "straight", "right" and "back". On the other hand, [88] uses the same as in naïve approach (such as left, right, straight) plus "slightly right", "sharp right" etc (idea derived from [55]) for defining turn instructions for road intersections. Figure 2.3 shows two types of qualitative representation of turn descriptions for road intersection. Naïve turn instructions can be ambiguous given the road network context. The authors suggest to use instructions such as sharp left or slightly right. In our work, we use naïve turn instructions to demonstrate that even sharing this information can present a privacy issue as the users actual route can be identified.

2.3 Privacy-preserving Techniques

LBSs provide answers to the queries based on user locations. For example, a traveller may want to know nearby points of interests such as a restaurant, a supermarket, an ATM machine or a foreign exchange. Though LBSs provide assistance to users, this assistance comes along with privacy threat. Since, a LBS provider knows the location of its users, a user's continuous access of LBSs enables the LBS provider to create a complete sketch of the user's movement with a high degree of spatial and temporal precision, possibly over an extended period of time. The LBS provider may infer private and sensitive information from this sketch. For example, if the user's location at a drug rehab centre then the LBS provider might infer that the user has had a drug addiction. Thus, preserving location privacy of a user is an important issue. Our work is highly related with the user location trajectory privacy. For this instance, we survey existing privacy preserving approaches for the user location trajectory privacy in this section. We at first discuss general data privacy principles in Section 2.3.1 and location privacy of a user in Section 2.3.2 because the idea of a user's location trajectory privacy derived from data privacy and location privacy. Finally, in Section 2.3.3, we review the existing principles of trajectory privacy of a user.

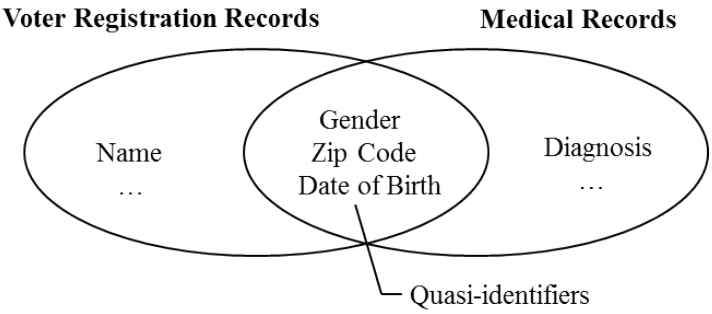

Figure 2.4: Using quasi-identifiers to join two micro-data sets [14]

2.3.1 Data Privacy

Data privacy, also called information privacy, is the aspect of information technology that deals with the ability an organization or individual has to determine what data in a computer system can be shared with third parties. Many organizations and agencies need to disclose individuals' personal data (e.g., gender, date of birth, zip code) for many practical purposes such as public health research, voter registration etc. Basically, these data or micro-data are stored in tables where each row corresponds to one individual and the identifiers (the identification of records in micro-data, i.e., name and social security number) must be removed for preserving the data privacy. However, combining such "*de-identified* micro-data with other released micro-data may still pose data privacy issues for individuals [78]. Thus, privacy protection of data is necessary. To illustrate the problem, data could have been *de-identified* by suppressing names and Social Security Numbers (SSNs) in order not to disclose the identities of the individuals to whom the data refer. However, values of other released attributes, such as zip code, gender and date of birth can also appear in some external table jointly with the individual identity, that can therefore allow them to be tracked. As stated in [14], a study estimated that 87% of the population of the United States can be uniquely identified by using the collection of non-identity attributes, i.e., gender, date of birth and 5-digit zip code [83]. In fact, those three attributes were used to link Massachusetts, USA voter registration records including name, gender, zip code and date of birth to *de-identified* medical data from Group Insurance Company including gender, zip code, date of birth and diagnosis to identify medical records of the governor of Massachusetts in the medical data [83], as illustrated in Figure 2.4. *Quasi-identifiers* and *equivalence class* refer to the attributes whose values taken together can potentially identify an individual record and a set of records that have

the same values for the quasi-identifiers in a released micro-data, respectively defined in [14]. In the following, we review a range of cognitive studies on protecting data privacy for an individual.

Overview of Data Privacy-preserving Approaches

Data privacy-preserving approaches have been designed to anonymize micro-data. Several privacy-preserving properties are proposed to limit disclosure of anonymized micro-data. For example, K-anonymity, (X, Y)-anonymity, MultiRelational K-anonymity, l-diversity, t-closeness, differential privacy etc. In this section, we discuss the most common data privacy preserving techniques, i.e., K-anonymity, l-diversity, t-closeness and differential privacy briefly. For the details of these and other data privacy principles, we refer the reader to the recent survey paper [31].

- ***K*-anonymity:** A release of data is said to have the K-anonymity property if the information for each person contained in the release cannot be distinguished from at least $K - 1$ individuals whose information also appear in the release. To prevent record linkage through quasi-identifier, [82] at first proposed the notion of K-anonymity. In the attack of record linkage, some value qid on quasi-identifier identifies a small number of records in the released table, called a group. If the victim's quasi-identifier matches the value qid, the victim is vulnerable to being linked to the small number of records in the group. In this case, the attacker faces only a small number of possibilities for the victim's record, and with the help of additional knowledge, there is a chance that the attacker could uniquely identify the victim's record from the group.

  K-anonymity requires each record to be indistinguishable with at least other $K1$ records with respect to the quasi-identifier, i.e., each equivalence class contains at least K records [58, 78, 82]. However, this method has two main limitations: firstly, a K-anonymized equivalence class suffers from a homogeneity attack if all records in the class have less than K values for the sensitive attribute (e.g., disease and salary). Secondly, it does not protect against attacks based on background knowledge.

- ***l*-diversity:** l-diversity property is proposed to ensure that an equivalence class must have at least l values for the sensitive attributes [63, 89]. l-diversity has the

limitation of implicitly assuming that each sensitive attribute takes values uniformly over its domain, that is, the frequencies of the various values of a confidential attribute are similar. When this is not the case, achieving l-diversity may cause a large data utility loss.

- ***t*-closeness:** t-closeness is defined that an equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the entire data set is no more than a threshold parameter t [61]. t-closeness uses the Earth Mover Distance (EMD) function to measure the closeness between two distributions of sensitive values, and requires the closeness to be within t. t-closeness has several limitations and weaknesses. First, it lacks the flexibility of specifying different protection levels for different sensitive values. Second, the EMD function is not suitable for preventing attribute linkage on numerical sensitive attributes. Third, enforcing t-closeness would greatly degrade the data utility because it requires the distribution of sensitive values to be the same in all quasi-identifier groups. This would significantly damage the correlation between quasi-identifiers and sensitive attributes.

- **Differential privacy:** Differential privacy, which allows a trusted LBS server to decide which answers, should be disclosed and how to add noise to answers to aggregation queries to avoid leaking information about any specific record in the database. It is a privacy definition that can be motivated in several ways [26]. If an adversary knows complete information about all individuals in the data except one, the output of the anonymization algorithm should not give the adversary too much additional information about the remaining individual. Alternatively, if one individual is considering lying about their data to a data collector (such as the Australian Census Bureau), the result of the anonymization algorithm will not be very different if the individual lied or not. Based on known definition in [27], formally we define it as follows:

  Definition 1. *Let R_A be a randomized algorithm, let S_O be the set of possible outputs of the algorithm, let P_O be a prior probability for the attributes of the remaining individual, and let $\in > 1$. The algorithm R_A satisfies $\in$-differential privacy if for all pairs of data sets (d_1, d_2) that differ in exactly one row,*

$$\forall S_O, \left| ln \frac{P_O(R_A(d_1) = S_O)}{P_O(R_A(d_2) = S_O)} \right| \leq \in$$

Differential privacy has the benefits that we do not have to assume that tuples are independent or that an adversary has a prior belief encapsulated by a probability distribution.

We distinguish two distinct types of attribute in privacy preserving mechanisms for protecting user location data; they are location data hiding based method and differential privacy based method. The former focuses on data removing of an individual location from the user data sets, for example, K-anonymity, l-diversity etc., for preserving user location data privacy; the latter focuses on adding different data such as hashing, sub-sampling and noise injection with user data sets to keep the data of individual users completely private. Differential privacy focuses on an aggregate results but our goal is to publish an individual record of location information. Thus, directional privacy does not directly apply into our work.

2.3.2 Location Privacy

Though LBSs provide many new opportunities, the ability to locate mobile users also presents new threats the intrusion of location privacy. Location privacy is defined as the ability to prevent unauthorized parties from learning one's current or past location [7]. Location privacy threats refer to the risk that an adversary can obtain unauthorized access to raw location data derived or computed location information by locating a transmitting device, hijacking the location transmission channel and identifying the subject using the device [41]. For example, location information can be used to spam users with unwanted advertisements or to learn about users' medical conditions, unpopular political or religious views. Inferences can be drawn from visits to clinics, doctor's offices, entertainment clubs or political events. Public location information can lead to physical harm, such as stalking or domestic abuse. So, the main concern is how to protect location privacy while accessing LBS system. Recently, considerable research [4, 7, 15, 22, 23, 34, 36, 37, 40, 48, 49, 52, 53, 54, 67, 85] interest has focused on preventing identity inference in location-based services. The main concern is to allow the mobile user to request services without compromising

his/her privacy. We classify these techniques into four main groups: cloaking, Pseudonym, cryptography and dummy.

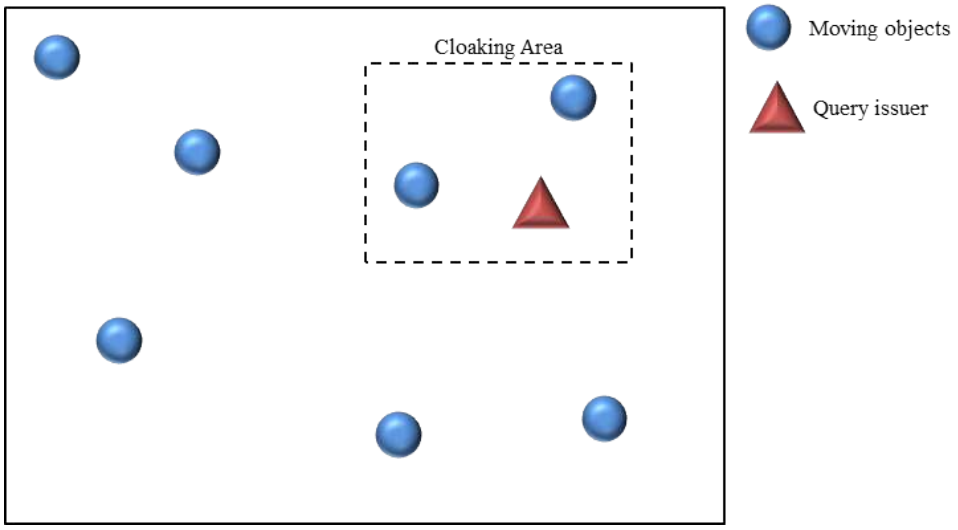

Figure 2.5: Cloaking based technique (based on K-anonymity) [46]

Cloaking Based Technique

In cloaking based technique, it generates blur area (circular or rectangular) that encloses actual query issuer user with other user based on his/her privacy requirement (e.g., K-anonymity, granularity metric etc). By this, the user can hide his/her location from adversary or LBS server. For example, in Figure 2.5, a user requests LBS with 3-anonymity through by trusted third party (Location cloaker) or group head (i.e., Distributed system), then location cloaker (LC) or group head (GH) makes the cloaking area and sends to the LBS for results set. A cloaking technique known as Interval Cloak introduced in [40], it calculates the cloaking region based on a Quad-tree method. The main idea is that it recursively partitions the space into four equally squared regions until a user fits in a quadrant to satisfy K-anonymity. However, this work suffers from several drawbacks. First, it assumes a system-wide static K value for all mobile users, which hinders the service quality for those mobile nodes whose privacy requirements can be satisfied using smaller K values. Furthermore, this assumption is unrealistic in practice as mobile users tend to have varying privacy protection requirements under different contexts and on different subjects. Second, their approach fails to provide any quality of service guarantees with respect to the sizes of the cloaking boxes produced. This is because, the Quad-tree

based algorithm anonymizes the messages by dividing the Quad-tree cells until the number of messages in each cell falls below K and by returning the previous quadrant for each cell as the spatial cloaking box of the messages under that cell.

To overcome the problems in [40], the CliqueCloak algorithm proposed in [34], it relies on the ability to locate a clique in a graph to perform location cloaking. However, This algorithm is expensive and performs poorly when K is large. In another approach, the possible locations are restricted to be nodes in a graph (e.g., a road network), and the cloaked region (or "obfuscation") is represented by a set of nodes [22, 23]. The cloaked region C' in Figure 2.6 is the set of blue nodes, one of them being q. The approaches that represent C' as a set quantify the privacy of C' by its cardinality.

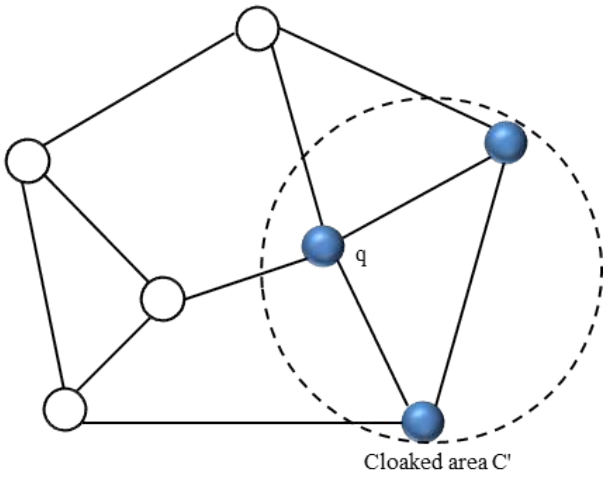

Figure 2.6: An example of obfuscation [92]

A novel location anonymization approach namely Casper proposed in [67], it addresses location anonymization using a pyramid data structure, similar to a Quad-tree and allows the system to quickly locate anonymized areas. Assume a user U asks a query and let c be the lowest-level cell of the Quad-tree where U lies. If c contains enough users (i.e., $\mid c \mid \geq K$), c becomes the cloaking region. Otherwise, the horizontal c_h and vertical c_v neighbors of c are retrieved. If $\mid c \cup c_h \mid \geq K$ or $\mid c \cup c_v \mid \geq K$, the corresponding union of cells becomes the cloaking region; otherwise, the anonymizer retrieves the parent of c and repeats this process recursively. However, Casper produces a large cloaking area and yields poor quality of services (QoS). Hilbert Cloak (HC) based cloaking method introduced in [52] that satisfies a reciprocity condition in every K-anonymous region. Reciprocity is an important property that is sufficient for spatial K-anonymity. For

example, A user u issues a query associated with anonymized set AS. The AS satisfies the reciprocity condition if there exists a set of users S_U lying in the AS such that $u \in S_U$ and every user $u \in S_U$ lies on the anonymous regions of all other users in S_U. Consider the Figure 2.7, where the user ids indicate their Hilbert order. For K-anonymity = 3, the users are grouped into 3 buckets (the last one contains 4 users). When any of U_1, U_2 or U_3 issues a query, HC returns the first bucket (shown shaded) as the AS; the minimum bounding rectangle (MBR) of that bucket becomes the 3 anonymizing set reciprocity $(3 - ASR)$. HC guarantees privacy for any distribution of user locations.

A grid-based cloaking method presented in [4] introduces l-diversity concept. Location l-diversity requires that every cloaked region containing K individuals contains at least l different point of interests (POIs). The degree of diversity is measured by counting the number of POI occurrences or types inside the cloaked region, while the different degree of sensitivity of those places is ignored. In essence, an l-diverse region "blurs in the crowd" the POIs where the person can stay, independently from the meaning of those places. The result is an excessive loss of position accuracy. CLOAKP2P approach proposed in [15], constructs a cloaking area by considering the neighbouring users of the query. Given cloaking region, the LBS returns the probability that each candidate result satisfies the query based on its location with respect to the cloaking region. On the other hand, it eliminates the anonymizer by organizing users in a Peer-to-Peer system. The querying user u searches his neighborhood until he finds $K - 1$ other peers, which are used to construct the cloaking region. However, u tends to be close to the centre of the cloaking region; therefore an attacker can identify u with high probability. It also fails to provide privacy protection when there are many users in a system.

Another Peer-to-Peer system to support distributed anonymization called PRIVE proposed in [36], where users form a cluster and maintain a hierarchical overlay network resembling a distributed B+ tree. However, it may suffer from a slow response time since the nodes in a root level constitute a potential bottleneck. X-Star based cloaking algorithm presented in [85], it achieves the optimal query processing cost and high privacy protection for the users. It groups neighbouring queries into a cloaking star structure and adjusts the result cloaking stars to satisfy the privacy requirement of users. In addition, it merges neighbouring stars into a super-star structure to meet the privacy requirements of each individual user. However, X-star has a low successful anonymization

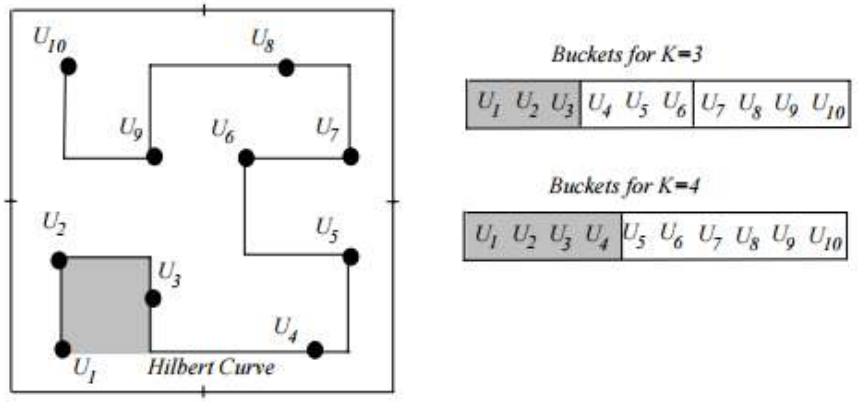

Figure 2.7: Example of Hilbert Cloak [52]

rate. For increasing high successful anonymization rate, [49] proposed a result-aware location cloaking approach called 2PASS, based on client-server model that minimizes the number of requested objects directly. It is based on a notion of Voronoi cells and each cell contains one object that is the nearest neighbour of any point in its cell. The user can fixed the cloaked area to the Voronoi cells of the nearest neighbour object, if he/she knows the Voronoi cells in advance. 2PASS is able to save the bandwidth usages and processes a K-nearest neighbor query in two steps. In the first step, the user requests the Voronoi cell information corresponding to the query. In the second step, it selects objects to request. Although 2PASS optimizes the bandwidth usages, it suffers from privacy attack.

Pseudonym Based Technique

Pseudonym combines the location and user identity called Mix-zones proposed in [7]. Mix-zones are recognized as an alternative and complementary approach to spatial cloaking based location privacy protection. The concept of "mix" has been applied to anonymous communication in a network. Mix-zones enable users to change pseudonyms such that the linking between the old and new pseudonyms is not revealed. The idea behind using pseudonyms instead of the real identities is to disassociate the exposure of location information from the actual one. The anonymity of a mix-zones enforcing that a set of users enter, change pseudonyms and exit a mix-zone in a way such that the mapping between their old and new pseudonyms is not revealed [7]. A mix-zone of K participants

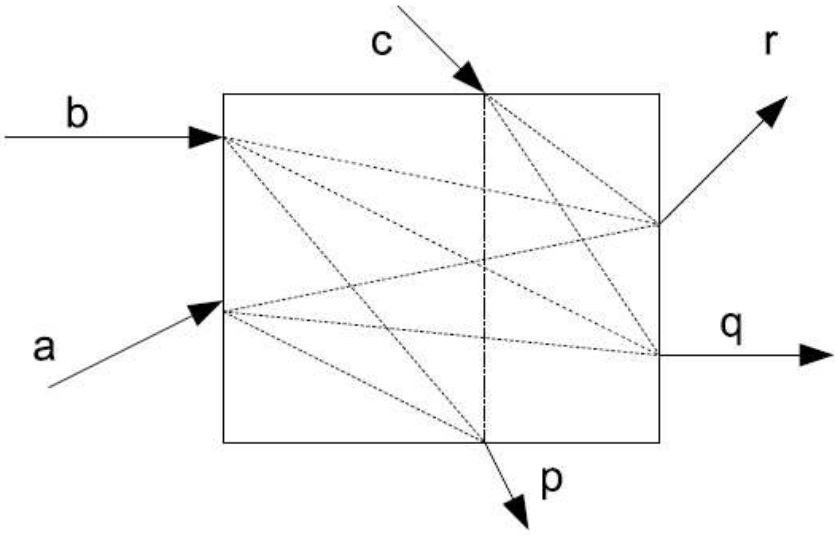

Figure 2.8: A Mix-zone with three users [70]

refers to a K-anonymization region in which users can change their pseudonyms such that the mapping between their old and new pseudonyms is not revealed. In a mix-zone, a set of K users enter in some order and change pseudonyms but none leave before all users enter the mix-zone. Inside the mix-zone, the users do not report their locations and they exit the mix-zone in an order different from their order of arrival, thus, providing unlinkability between their entering and exiting events [70]. A set of users S_U is said to be K-anonymized in a mix-zone Z iff:

- The set S_U consists K or more users, i.e., $\mid S_U \mid \geq K$.
- All users in S_U must enter the mix-zone Z before any user $i \epsilon S_U$ exits. Thus, there exists a point in time where all K users of S_U are inside the zone.
- Each user $i \epsilon S_U$, entering the mix-zone Z through an entry point $e_i \epsilon E$ and leaving at an exit point $o_i \epsilon O$, spends a completely random duration of time inside.
- The probability of transition between any point of entry to any point of exit follows a uniform distribution. i.e., a user entering through an entry point, $e_i \epsilon E$, is equally likely to exit in any of the exit points, $o_i \epsilon O$.

Figure 2.8 depicts a mix-zone with three users entering with pseudonyms a, b and c and exiting with new pseudonyms, x, y and z. Given any user exiting with a new

pseudonym, the adversary has equal probability of associating it with each of the old pseudonyms a, b and c and thus this example mix-zone provides an anonymity of $K = 3$.

Cryptography Based Technique

Cryptography based approaches [37, 53] allow users to access LBSs without revealing their locations to the LBS provider (LSP). Although the architecture of both of these approaches eliminates an intermediary trusted server, these approaches incur cryptographic overheads and cannot use existing spatial indexing techniques to store data on the server.

In [53], the data space is encrypted from two dimensions to one dimension space using Hilbert curves by a trusted third party and the transformed space can be decrypted using a key, which is not known to the LSP and the users. This approach assumes that the user's tamper-resistance device stores the key and encrypts the location before forwarding a query to the LSP. The LSP evaluates the K-nearest neighbour query in the transformed space and returns the encrypted data objects which are again decrypted to the original space in the user device. Since the query for k nearest data objects is evaluated in an encrypted space which might not always preserve the proximity relations of the original space, the returned data objects may not be the actual answers. In this approach, the privacy could be violated if any of these tamper-resistance devices gets compromised.

[37] uses private information retrieval (PIR) protocols to retrieve the approximate and exact nearest data object without disclosing a user's location. In this approach, the space is divided into grid cells and the data objects associated each cell are stored in a format required by PIR protocols. The user provide her encrypted cell, where she is located, using PIR, and determines the nearest data object from the retrieved encrypted data objects. Although this approach ensures privacy, it incurs a high pre-processing overhead compared to spatial cloaking. Moreover, this approach only supports queries for the nearest data object.

Dummy Based Technique

Dummy based cloaking technique presented in [7, 48, 54], it generates false user locations called dummies and combine them with the actual user location into the request. The purpose of these fake locations is to increase the uncertainty of the adversary about the users' real movements. For example, in Figure 2.9, d_1, d_2, d_3, d_4 are dummy locations, and

the cloaked region is $C^{'} = \{u, d_1, d_2, d_3, d_4\}$. However, the server may infer the actual user location from dummies after monitoring long-term movement patterns of the user. [48] proposes dummy based technique but it shows poor performance in a real environment. [7] improves this technique by generating consistent movement patterns for dummies in a long run. Scattered dummies generated around the user based on the user's current location and previous dummies locations in [54] for improving the performance of [7]. It also consider dummies locations according to the real road information.

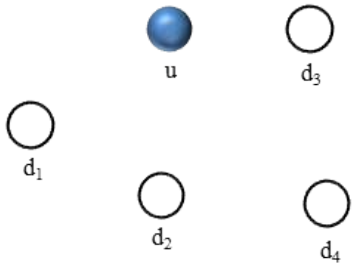

Figure 2.9: Dummy based technique [92]

2.3.3 Trajectory Privacy

Though, a plethora of work has been done on the hiding of location information of users, little attention has been paid to the users who are accessing LBS system during traveling time on the road. Furthermore, the condition of the road network, e.g., the network topology has significant impact over the query evaluation and communication efficiency, which should be a critical concern for developing location privatization solutions. For instance, the complexity of computing the network distance of two points, a most fundamental operation in location-based query processing, varies considerably with the underlying network structure. In this section, we will discuss some of the related works which has been done in order to preserve individual's location privacy in a continuous scenario i.e., privacy in trajectory data.

A trajectory is a path reported by a moving object in the geographical space. A spatial trajectory is represented by a set of n time-ordered points where each point consists of a geo-spatial coordinate set (x_i, y_i) and time-stamp (t_i). Trajectory data basically compact forms of highly sensitive information of user (e.g., location points) and therefore, there is a high probability to uncover the information due to publishing it carelessly or even an incomplete part of it can cause serious privacy issues [31]. This types of spatial and

temporal attributes of a location trajectory can be considered as powerful quasi-identifiers that can be linked to various other kinds of physical data objects [31, 68]. Based on the example in [14], a hospital releases a trajectory data set of its patients to a third-party research institute for analysis, as shown in Table 2.1. The released trajectory data set does not contain any explicit identifiers, such as patient name, but it contains a sensitive attribute (i.e., disease). Each record with a unique random ID, RID, corresponds to an individual,e.g., the record with $RID = 1$ means a patient visited locations $(1, 2)$, $(1, 5)$ and $(3, 6)$ at timestamps 1, 4 and 7, respectively. Suppose that an adversary knows that a patient of the hospital, Adam, visited locations $(1, 5)$ and $(3, 6)$ at timestamps 4 and 7, respectively. Since only the trajectory record with $RID = 1$ satisfies such spatial and temporal attributes, the adversary can infer that Adam has High pressure. This example shows that publishing "de-identified" trajectory data can still cause serious privacy threats.

Table 2.1: Patient Trajectory Records

RID	**Trajectory**	**Affliction**	...
1	(1, 2, 1)-(1, 5, 4)-(3, 6, 7)	High pressure	...
2	(3, 2, 5)-(6, 5, 4)-(7, 6, 3)	High pressure	...
3	(10, 2, 5)-(11, 5, 8)-(13, 6, 9)	Chest pain	...
4	(7, 6, 3)-(6, 7, 4)-(6, 10, 6)	HIV	...

In case of facing any privacy risk, the location information of that part of the trajectory is suppressed in a way to render a secure, publishable database. Likewise, [2, 50, 68] have been proposed anonymizing methods for preserving privacy in trajectories. In [68], it employs the concept of K-anonymity in order to make a certain trajectory indistinguishable from $K-1$ other trajectories via a generalization-based algorithm; thus, privacy is ensured by releasing only a randomly generated set of representative trajectories. In [2], the algorithm though, approaches K-anonymity from a different angle. It proposed cluster-based algorithm utilizes an uncertainty threshold, which is inherent to trajectory data, in order to group K co-localized trajectories within the same time period to form a K-anonymized aggregate trajectory. All these works though, focus on changing the

entire trajectory as a means of preserving privacy; this leads to a more-than-necessary distorted database, which considerably decreases the quality of querying and ruins mining opportunities.

On the other hand, some new studies have focused on destination points along a path as the most sensitive parts of a trajectory and proposed some solutions to protect these points. In [50], the authors tried to preserve sensitive destination points by generalizing them into l-diverse zones, i.e., areas that contain at least l distinct place types such as restaurant, hospital, park and etc. The problem is in [50], it treated all the stop points equally and tried to generalize the trajectory accordingly, while in reality, some places like hospitals should be prioritized in terms of sensitivity compared with, for instance, shopping malls. In this section, we discuss four privacy-preserving techniques for trajectory privacy, namely, spatial cloaking, vehicular mix-zones, path confusion and dummy.

Spatial Cloaking

Mobile users have to reveal their locations to the LBS service provider in a periodic manner to obtain continuous LBS. By applying a snapshot spatial cloaking technique (e.g., [4, 10, 22, 34, 35, 36, 40] to each user location independently cannot ensure k-anonymity for a user location trajectory [14]. Thus, new spatial cloaking techniques based on either real-time or historical user trajectories are designed to protect user location trajectories. Similar to snapshot spatial cloaking techniques, a fully-trusted third party, usually termed location anonymizer, is placed between mobile users and database servers. The location anonymizer is responsible for collecting users' locations and blurring their locations into cloaked spatial regions that satisfy the user-specified k-anonymity level and/or minimum spatial region area. Since spatial cloaking techniques do not change user identities. In the following sections, we will discuss three main kinds of spatial cloaking techniques over user trajectories, namely, group based, distortion based, and prediction based approaches [14].

- **Group based Approach:** The group-based algorithm is proposed to use real time location trajectory data to protect trajectory privacy for continuous location-based queries in [12]. The main idea is that a querying user u forms a group with other $K-1$ nearby users. It blurs u's location into a spatial area that contains all the group members as a cloaked spatial area before the algorithm issues u's location-

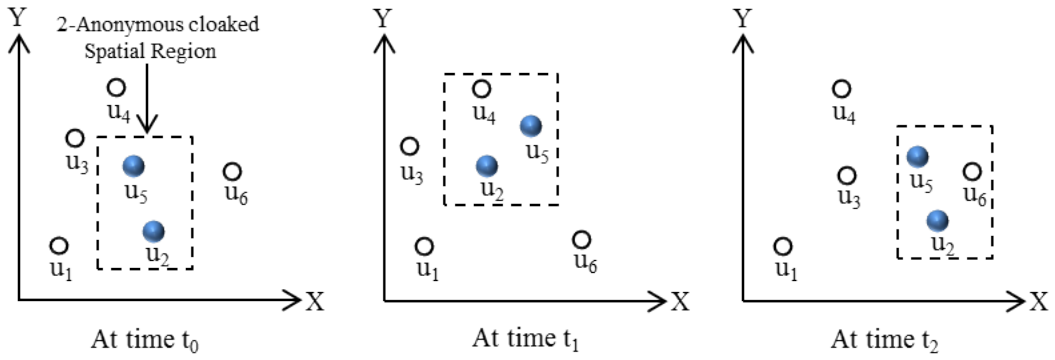

Figure 2.10: Group based spatial cloaking approach [13]

based query or reports u's location to the database server. For an example, let consider the Figure 2.10, a user u_2 issues a continuous location-based query at time t_0 requires its location to be K-anonymized, where $K = 2$. At time t_0, a location anonymizer forms a group of users u_2, and u_5, so that u_2's cloaked spatial region contains all these group members, as represented by a rectangle in the left Figure 2.10. The location anonymizer sends u_2's query with its cloaked spatial region to a database server. At later times t_1 and t_2, when u_2 reports its new location to the location anonymizer, a new cloaked spatial region that contains the group members is formed, as shown in middle and right figures in the Figure 2.10 respectively. The drawbacks of this approach are that users not issuing any query have to report their locations to the location anonymizer and the cloaked spatial area would become very large after a long time period. Such a large cloaked spatial area may incur high computational overhead at the database server and results in many candidate answer objects returned from the database server to the location anonymizer.

- **Distortion based Approach:** Distortion based cloaking approach proposed in [71], it takes users' moving speed and direction as affecting factors and a distortion function is defined to measure the temporal query distortion of a cluster in continuous queries. The main objective of the distortion based approach is to overcome the drawbacks of the group-based approach. For an example, in Figure 2.11, three users u_1, u_2 and u_3 proceed their continuous location-based queries at time t_0 form a 3-Anonymous cloaked area $C_1^{'}$ and their queries expire at time t_n. Their cloaked spatial region $C_1^{'}$ at time t_0 is a minimum bounding rectangle of the cloaking set, as represented by a rectangle in the figure. Assume (x_i, y_i) and $(x_i^{'}, y_i^{'})$ be the left

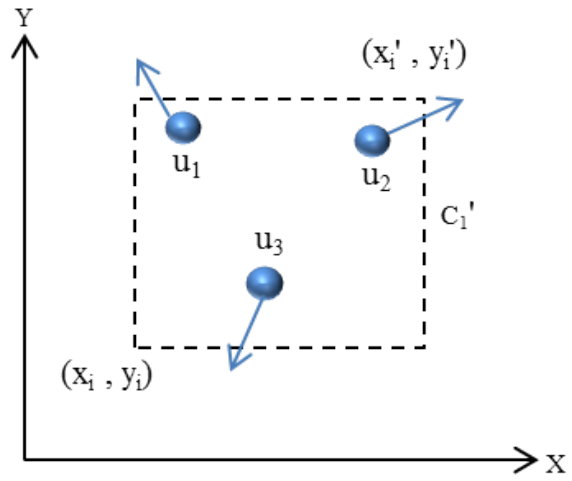

Figure 2.11: Cloaked spatial area C_1' at time t_0 in distortion based spatial cloaking approach [13]

bottom and right top nodes of C_i' at time t_i, respectively. The distortion for their queries with C_i' at time t_i is defined as:

$$\Delta(C_i') = \frac{(x_i, y_i) + (x_i', y_i')}{M_h + M_w}$$

Where M_h and M_w are the height and width of the minimum bounding rectangle of the entire system, respectively.

- **Prediction based Approach:** Prediction based approach introduced in [90], it ensures K-anonymity by utilizing individuals' historical footprints, instead of their real time locations. A footprint is defined as a user's location collected at some point of time. It considers a trusted location anonymizer is placed between users and LBS service providers to collect users' footprints. The location anonymizer cloaks the user's predicted trajectory i.e., a sequence of expected footprints with $K1$ historical trajectories collected from other users. Consider a scenario in Figure 2.12, a user u_1 wants to subscribe continuous LBS from a service provider. u_1's predicted trajectory i.e., expected footprints are p_1 to p_6 represented by black circles. If u_1's desired anonymity level is $k = 2$, the location anonymizer finds historical trajectories from one other user, u_2. Then, each u_1's expected footprint p_i $(1 \leq i \leq 5)$ is cloaked with at least one unique footprint of u_2's trajectories to form a cloaked spatial region C_i. The sequence of such cloaked spatial regions set the K-anonymized trajectory for u_1.

This approach provides better resolutions for K-anonymized trajectories. However, it may suffer from an observation attack when a user stay alone in a cloaked area or less than K users located in a cloaked area at its associated time-stamp.

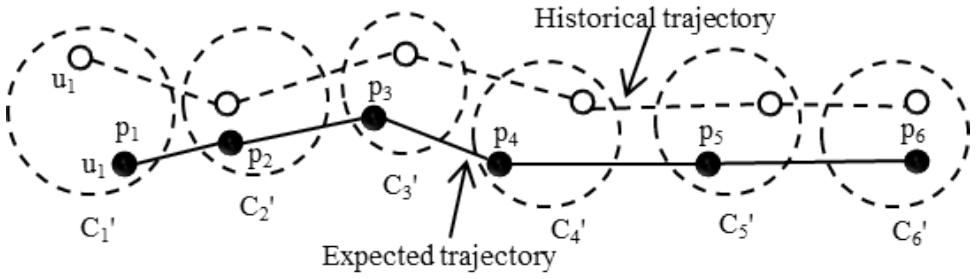

Figure 2.12: An example of prediction based spatial cloaking approach [13]

Path Confusion

Path confusion technique has been introduced in [42], it aims to predict the position of a target vehicle based on the last known speed and direction information and then decide which next location sample (or the one with the highest probability if there are multiple candidate location samples) to link to the same vehicle through Maximum Likelihood Detection. It avoids linking consecutive location samples to individual vehicles through target tracking algorithms with high certainty. In this method, The degree of privacy is defined as the "time-to-confusion", i.e., the tracking time between two location samples where an adversary could not determine the next sample with sufficient tracking certainty. However, path confusion technique only supported by the specific type of LBSs where LBSs do not require any consistent user identity or even any user identity, such as "send discount information for each menu to users within 2km from my restaurant".

Dummy Trajectories

For preserving trajectory privacy of a user, [54] at first proposed dummy based trajectory privacy preserving approach. It uses fake location trajectories, called dummies that are generating by a user without relying on a trusted third party to perform anonymization. In [93], for a given user real spatial trajectory T_r and a set of user generated dummies T_d, the degree of privacy protection for the real trajectory is measured by three parameters. They are snapshot disclosure (SD), trajectory disclosure (TD) and distance

deviation (DD), respectively. SD refers the average probability of successfully inferring each true location sample in T_r. TD defines as the probability of successfully identifying the true trajectory among all possible trajectories. And DD defines as the average distance between the i-th location samples of T_r and each T_d. Dummy based anonymization algorithm takes a real trajectory T_r and the three user-specified parameters (i.i., SD, TD, and DD in a privacy profile) as an input. It then incrementally uses DD to find a set of candidate dummies and select one with the best matching to SD and TD until it finds a set of trajectories (including T_r and selected dummies) that satisfies all the above parameters.

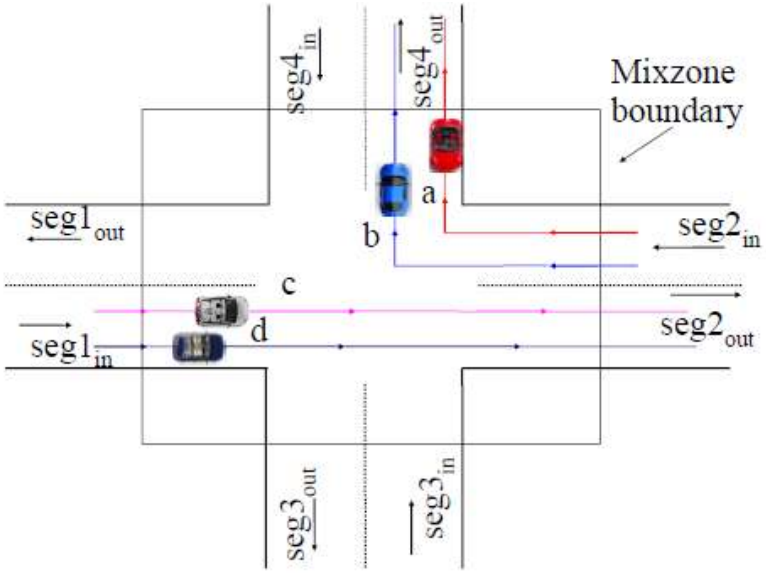

Figure 2.13: A vehicular Mix-zone [70]

Vehicular Mix-zones

The idea of vehicular mix-zones is derived from mix-zones [7]. However, the anonymization method of mix-zones could not be effective on vehicular mix-zones because mix-zones designed for the Euclidean space are not secure enough to protect trajectory privacy in road networks [30, 69]. Vehicle movements are restricted in a road network due to having spatial and temporal factors such as physical roads, directions, speed limits etc. If mix-zones construct in road network an adversary could able to utilize these road restriction's knowledge to identify users' identity. Figure 2.13 depicts an example of vehicular mix-zones, a mix-zone (i.e., rectangle) is placed on the intersection of four road segments

S_1, S_2, S_3 and S_4. Users a and b enter the road intersection from S_2 and turn on to S_4. Users c and d enter from S_1 and leave on S_2. When user a and b exit the mix-zone on S_4 with their new pseudonyms, let say α and β, the attacker tries to map their new pseudonyms α and β to some of the old pseudonyms a, b, c, and d of the same users. The new pseudonym α is more likely to be mapped to two of the old pseudonyms, a or b, than the other pseudonyms because users a and b entered the mix-zone well ahead of users c and d and it is thus less probable for c and d to leave the mix-zone before users a and b given the speed and trajectory of travel. Here, the limited randomness on the time spent inside a road network mix-zone introduces more challenges to construct efficient mix-zones.

To overcome the problem, vehicular mix-zones has been proposed in [69] and it considers the following properties for a K-anonymized location to a set of users S_A:

- There are K or more users in the cloaked area $C^{'}$.
- Given any two users $(i, j) \in C^{'}$ and assuming i existing at time t, the pairwise entropy after timing attack[1] should satisfy the condition: $H_{pair}(i, j) \geq \alpha$.
- For any two users $(i, j) \in C^{'}$, the pairwise entropy after transition attack[2] should satisfy the condition: $H_{pair}(i, j) \geq \beta$.

Consider the Figure 2.14, The length of each mix-zone on an outgoing segment is determined based on the average speed of the road segment, the time window, and the minimum pairwise entropy threshold. The shaded area in the mix-zone is to ensure that an adversary cannot infer the vehicle movement direction (e.g., turn left or go straight in this example). The pairwise entropy is computed for every pair of users a and b in an anonymized set $C^{'}$ by considering a and b to be the only members in $C^{'}$ and determining the linkability between their old and new pseudonyms.

In summary, the overall idea of the above existing works is to preserve users' location trajectory privacy by removing trajectory information from the raw trajectory data that

[1]Timing attack refers to the attack that happens due to the close observation of the attacker from the time of entry and the time of exit for each user entering and exiting the mix-zone. For details we refer to the reader in [69].

[2]Transition attack occurs when the attacker estimates the transition probability for each possible turn in the intersection based on previous observations. For details see [69].

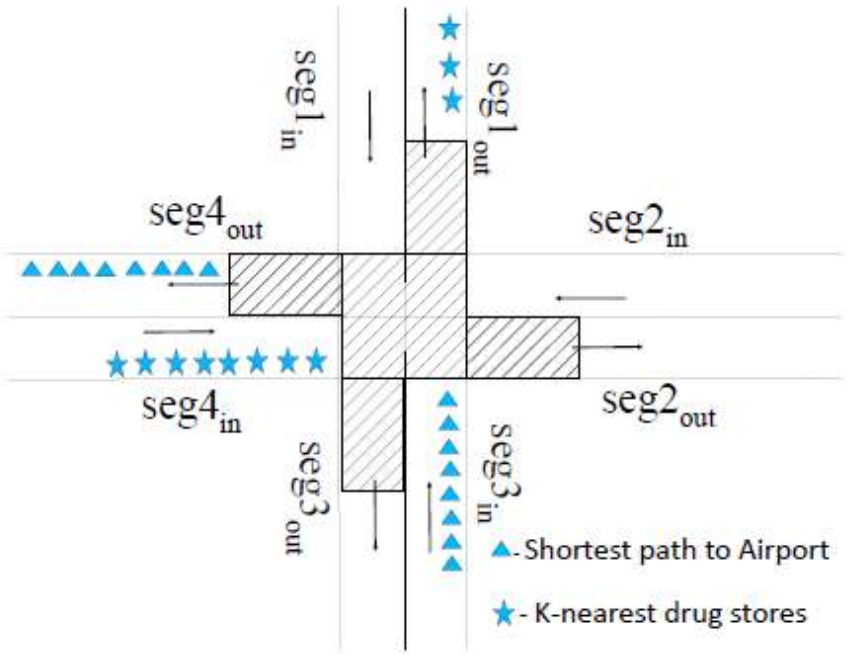

Figure 2.14: Non-rectangular, adaptive vehicular Mix-zone [70]

users reveal while they are using LBSs. However, the raw trajectory data sets contain some location information at the end.

2.4 Summary

In this chapter, we have discussed the importance of navigation services and the overall existing methods for preserving user location privacy in location based services (LBSs). The discussion includes problems with navigation instructions and relevant cognitive approaches, different privacy models and techniques to compute the user's cloaked area. We break down the importance of preserving users' location privacy in LBSs into three main areas: data privacy, location privacy, and trajectory privacy, and then survey existing privacy preserving techniques along these areas. Data privacy is the information privacy of an individual that deals with the ability an organization or individual has to determine what data in a computer system can be shared with third parties. Location and trajectory privacy are the ability to prevent unauthorized parties from learning individual's current or past location and current or past trip or parts of it, respectively. We have observed from the above survey, range of studies are pointed out the importance of preserving privacy of a user location trajectory and for this users' location information

have removed assuming privacy being preserved. Nevertheless, relevant information of the trajectory such as turn instructions, distance and so forth retain with data sets that may link to identify the user location at the end.

Chapter 3

Background

In the previous chapter, we underlined the importance of preserving the user's trajectory privacy whilst using LBSs. In this chapter, we study the problem of finding the shortest path between two vertices (nodes) in a graph. Shortest path problems are inevitable in road network applications such as drive guiding systems that include navigation directions. Basic concepts of network analysis in connection with traffic issues are explored. City traffic conditions change frequently and using requests are made therefore, the guiding system needs to find solutions rapidly. The above problems can be improved by using a shortest path computation algorithm. The shortest path is the path of minimum weight (can be based on distance, time etc) connecting two specified locations.

The shortest path problem is frequently encountered in our daily life. Let us consider the example in Figure 3.1, Alice wishes to go to the "Royal Melbourne Hospital" from her current location "Flinders Street Railway Station", for a clinic appointment so she needs to reach the hospital on time. Finding the shortest path is highly desirable for Alice in this circumstance. Basically, our proposed work is based on the road network and for this reason the shortest path problem needs to be addressed first. Road networks are often represented by graphs. The shortest path problem is to find a path between two nodes on a given graph, such that the sum of the weights on its constituent edges is minimised. Numerous shortest path computation approaches [6, 20, 28, 45, 51, 86] have been proposed for solving this important issue. In this chapter, we discuss five different solutions in the context of computing shortest paths between two locations and they are as follows: 1) Dijkstra's Algorithm, 2) Bellman-Ford Algorithm, 3) Floyd-Warshall Algorithm, 4) Johnson Algorithm and 5) A^*-Search Algorithm, respectively.

The remaining part of this chapter incorporates six sections as follows: we present the basic idea of graph representation in Section 3.1. We then discuss the above mentioned five shortest path problem algorithms from Section 3.2 to Section 3.6. Finally, in Section 3.7, we discuss the analysis results of all the above shortest path computation algorithms.

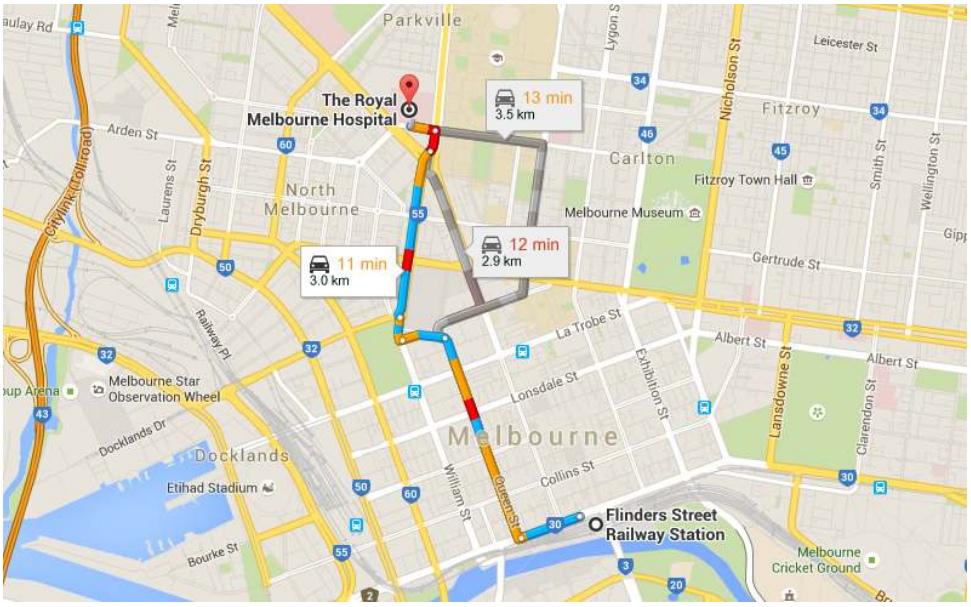

Figure 3.1: Importance of choosing shortest path in the road network

3.1 Graph

Graph G represents the road network. It consists of a set of vertices or nodes (V_G) and a set of edges (E_G) that connect a pair of nodes. When an edge connects two nodes, we say that the nodes are adjacent to one another and that the edge is incident on both nodes. Each edge of G has an associated numerical value, called a weight(w). Usually, the edge weights are non-negative integers. The weight of an edge is often referred to as the length or distance of the edge. The degree of a node ($d_G(n)$) is the number of edges incident on it. A path (P) in G from a node n_1 to a node n_k is a sequence of edges (n_1,n_2), (n_2,n_3),...,(n_{k-1}, n_k). The length $w(P)$ of a path P is the sum of the weights of the edges that belong to P. We say that one node is connected to another if there exists a path that contains both of them. $P^* = (s, ..., t)$ is a *shortest path* if there is no path P' from s to t such that $w(P') < w(P^*)$. G is connected if there is a path from every node to every other node. The *distance* $d(s, t)$ from s to t in G is the length of a shortest path from s to t or ∞, if there is no path from s to t. There are different types of graphs

(e.g., undirected, directed, simple, multigraph etc.) that have different formal definitions, depending on the kinds of edges that are allowed. Basically, the first two types of graphs from the above example are used to represent networks of communication. An undirected graph is a graph consisting of a set of objects (called vertices or nodes) that are connected together, where all the edges are bidirectional. In contrast, a graph where the edges point in a direction is called a directed graph.

3.2 Dijkstra's Algorithm

Dijkstra's algorithm [20] is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs. One of the main reasons for the popularity of Dijkstra's algorithm is that it is one of the most important and useful algorithms available for generating optimal solutions to a large class of shortest path problems. We choose Dijkstra's algorithm for computing shortest paths in the road network represented by a graph with non-negative edge path costs. Starting with the source node s as root, Dijkstra's algorithm grows a shortest path tree that contains shortest paths from s to all other nodes. During this process, each node of the graph is unreached, reached, or settled. A node that already belongs to the tree is settled. If a node u is settled, a shortest path P^* from s to u is found and the distance $d(s, u) = w(P^*)$ is known. A node that is adjacent to a settled node is reached. Note that a settled node is also reached. If a node u is reached, a path P from s to u, which might not be the shortest one, is found and a tentative distance $\delta(u) = w(P)$ is known. A node u that is not reached is unreached; for such a node, we have $\delta(u) = \infty$. The nodes that are reached but not settled are managed in a set S and the distance is updated for it's adjacent node. When multiple distances are recorded in S, the smallest one is chosen by it. When all of the nodes have been recorded in S and all paths in the diagram have been visited this algorithm terminates.

Initially, s is inserted into the set S with the tentative distance 0. Thus, s is reached and all other nodes are unreached. While S is not empty, the node u with the smallest tentative distance is removed and added to the shortest path tree, i.e., u becomes settled. Furthermore, u's outgoing edges are relaxed:

- If an edge (u, v) leads to an unreached node v, v is added to the S; now, v is

reached;

- If an edge (u, v) leads to a queued node v, v's key in S is updated, if the length of the path from s via u to v is less than v's old key;
- If an edge (u, v) leads to a settled node v, it is ignored.

Figure 3.2 depicts an example to show how the Dijkstra's algorithm works to find the shortest path in the road network. In this figure, circles correspond to location points or points of interest (POI), i.e., cafes for example, and solid lines represent the link between two location points. Suppose we want to find the shortest path from the location point n_1 to the location point n_{12}. The process is as follows. Update all n_1's adjacent nodes as its distance to n_1. Set $n_2(1)$, $n_6(3)$. Then from n_2, update all n_2's adjacent nodes as its distance to n_1 are $n_3(3)$, $n_7(2)$. Then from n_6, update all n_6's adjacent nodes as its distance to n_1 are $n_7(5)$, $n_{11}(13)$ (ignore 5) . Then from n_7, update all adjacent nodes as its distance to n_1 are $n_8(6)$, $n_{12}(3)$, $n_6(4)$. Update all distances (ignore 4). Finally, from n_{11}, update n_{12} as $13 + 3 = 16$ (ignore 16 as it is larger than 3). The shortest path is $n_1 \longrightarrow n_2 \longrightarrow n_7 \longrightarrow n_{12}$, and the distance is 3.

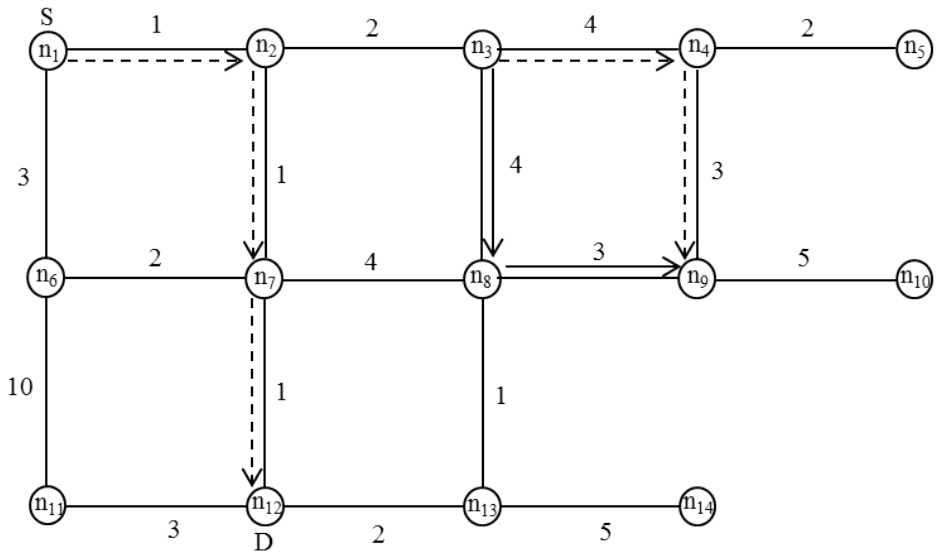

Figure 3.2: An example of Dijkstra's algorithm applied to a road network

Dijkstra's algorithm has been classified based on efficiency compared with the original version of Dijkstra's algorithm (described above), i.e., how quickly they can compute shortest path distances. The main focus of modified work on shortest paths has been

how to reduce priority queue operations. The basic Dijkstra's algorithm runs in $O(|V|^2)$. This bound has been improved several times, e.g., to $O(|E|\,log\,|V|)$ using binary heaps [3], $O(|E| + |V|\,log\,|V|)$ using Fibonacci heaps [29] etc. We adopt the basic Dijkstra's algorithm in our proposed work because our main intention is to show the extent to which a users' location privacy attack can occur while they choose their path, not to improve the effectiveness of the shortest path algorithm.

We also compute all possible shortest paths in a network by using Dijkstra's algorithm; but, we observe that Dijkstra's algorithm is not feasible for identifying the most suitable shortest path when the shortest paths in a graph are not unique. Figure 3.2 depicts the problem that arises when Dijkstra's algorithm finds more than one shortest path between two location points. For example, two shortest paths are found from the location point n_3 to n_9 (assigned by black solid and dashed arrows). In this situation, randomly any of between two shortest paths is chosen by Dijkstra's algorithm. Let us say that the yellow shortest path is chosen and stored in the shortest path library; however, the green shortest path is the more suitable shortest path from n_3 to n_9. We modify Dijkstra's algorithm by employing the A^*-Search algorithm (we will describe later in this chapter) in the case of finding the most suitable shortest path among the multiple shortest paths between two locations in the road network.

3.3 Bellman-Ford Algorithm

Bellman-Ford algorithm [6] is an algorithm that computes shortest paths from a single source node to all of the other nodes in a weighted directed graph. Basically, the Bellman-Ford algorithm is designed for directed graphs, but if the graph is undirected, substitution are made i.e., every edge (u, v) is replaced with two directed edges (u, v) and (v, u), both with weight $w(u, v)$. It is slower than Dijkstra's algorithm for the same problem but can be used for graphs with negative edge weights, which does not apply to the problems considered in this thesis. The algorithm requires that the graph does not contain any cycles of negative length but if it does, the algorithm can detect them.

The Bellman-Ford algorithm is based on the relaxation operation. The relaxation procedure takes two nodes as arguments and an edge connecting these nodes. If the distance from the source to the first node (for example A) plus the edge length is less

than the distance to the second node, then the first node is denoted as the predecessor of the second node and the distance to the second node is recalculated ($distance(A) + edge.length$). Otherwise no changes are applied. The path from the source node to any other node can be at maximum $|V| - 1$ edges long, provided there is no cycle of negative length; hence if we perform the relaxation operation $|V| - 1$ for all nodes, the algorithm will find all shortest paths. We verify the output by running the relaxation once more; if some edges are relaxed, the algorithm will contain a cycle of negative length and the output is invalid. Otherwise the output is valid and the algorithm can return the shortest path tree.

Figure 3.3 depicts the example of choosing the shortest path between two locations in the road network (where paths are following directions) by using the Bellman-Ford algorithm. For example, we wish to find the shortest path from location point n_1 to location point n_{12}. The Bellman-Ford algorithm at first checks all adjacent nodes from the source node n_1 and updates their distances. Set $n_2(1)$ and $n_6(3)$ as they are adjacent nodes of n_1. From n_2, update n_2's adjacent node as its distance to n_1 is $n_6(2)$. Then from n_6, update n_6's adjacent node as its distance to n_1 is $n_{11}(5)$. Then from n_7, update all adjacent nodes as its distance to n_1 are $n_6(0)$, $n_{12}(3)$, $n_8(6)$. Update all distances (ignore 3 and 5). From n_{11}, update n_{11}'s adjacent node as its distance to n_1 is $n_{12}(1)$ and ignore 3 as it is larger than 1. The resulting shortest path is $n_1 \longrightarrow n_2 \longrightarrow n_7 \longrightarrow n_6 \longrightarrow n_{11} \longrightarrow n_{12}$, and the distance is 1.

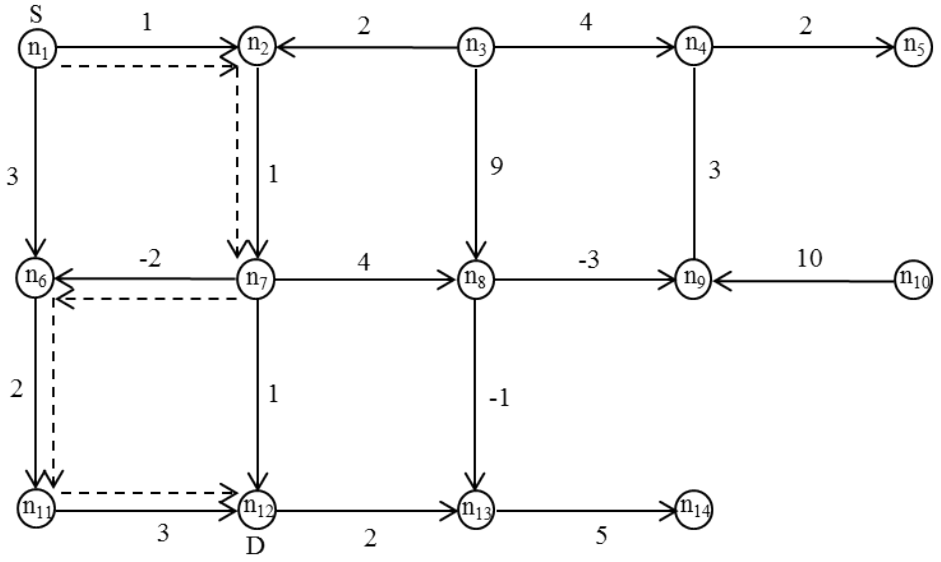

Figure 3.3: An example of the Bellman-Ford algorithm

The time complexity of the Bellman-Ford algorithm takes $O(|V|\,|E|)$ time where $|V|$ and $|E|$ are the number of nodes and edges. This time complexity is the best time bound for the single source shortest path problem [11].

3.4 Floyd-Warshall Algorithm

The Floyd-Warshall algorithm was developed independently by Floyd and Warshall in 1962 [28, 86]. Instead of computing a path from a given start node to all other nodes (or a single destination node), all shortest paths, from each node to all others, are computed within a single loop. This algorithm works on directed graphs with positive or negative edge weights (but with no negative cycle). The characteristics of the Floyd-Warshall algorithm are not appropriate for the problem considered in our work.

The Floyd-Warshall algorithm uses a matrix of lengths d_0 as its input. If there is an edge between nodes i and j, then the matrix d_0 contains its length at the corresponding coordinates. The diagonal of the matrix contains only zeros. If there is no edge between edges i and j, than the position (i, j) contains positive infinity. In other words, the matrix represents lengths of all paths between nodes that do not contain any intermediate nodes. In each iteration of the Floyd-Warshall algorithm this matrix is recalculated, so it contains lengths of paths among all pairs of nodes using a gradually enlarging set of intermediate nodes. The matrix d_1, which is created by the first iteration of the procedure, contains paths among all nodes using exactly one (predefined) intermediate node. d_2 contains lengths using two predefined intermediate nodes. Finally the matrix d_n uses n intermediate nodes. This transformation can be described using the following recurrent formula:

$$d_{ij}^{n} = min(d_{ij}^{n-1}, d_{ik}^{n-1} + d_{kj}^{n-1})$$

Because this transformation never rewrites elements, which are to be used to calculate the new matrix, the same matrix can be used for both d^i and d^{i+1}. The time complexity of the Floyd-Warshall algorithm is $O(|V|^3)$, where $|V|$ is the number of nodes in the graph. An example is shown in figure 3.4.

Another output matrix p (matrix of predecessors), has been constructed during transformations of matrix d_k in order to return shortest paths among all pairs of nodes. The matrix can be read as follows: if we want to reconstruct the shortest path between nodes i

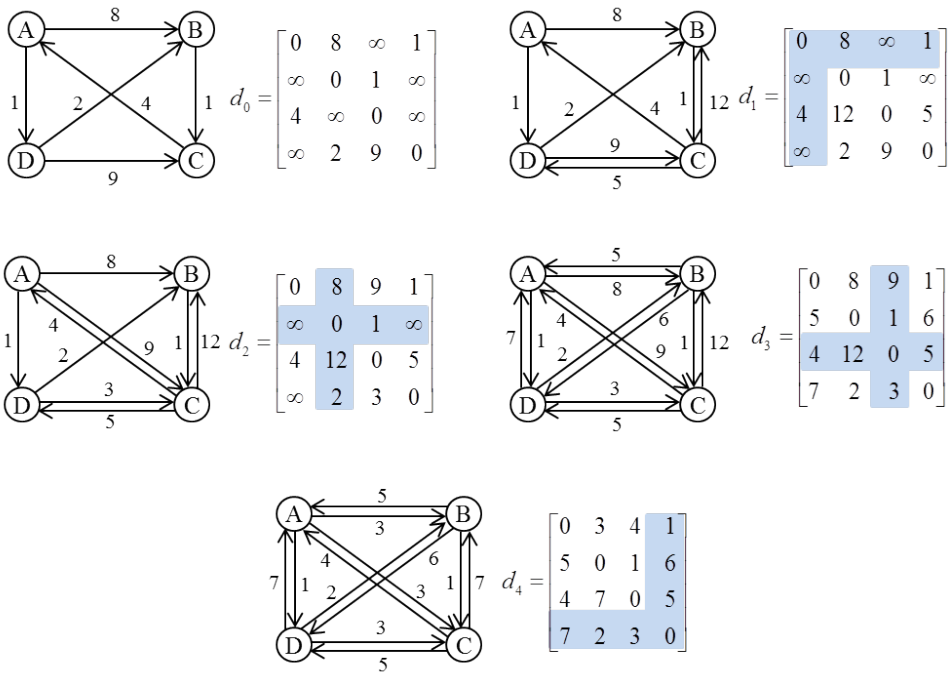

Figure 3.4: An example of the Floyd-Warshall algorithm

and j, we look at the element at corresponding coordinates. If its value is 0, than there is no path between these nodes, otherwise the value of the element denotes the predecessor of j on the path from i to j. This procedure repeats while the preceding node is not equal to i.

$$p_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \text{ or } d_{ij} = \infty \\ i & \text{in all other cases.} \end{cases}$$

$$p_{ij}^{(n)} = \begin{cases} p_{ij}^{(n-1)} & \text{if } d_{ij}^{(n-1)} \leq d_{ik}^{(n-1)} + d_{kj}^{(n-1)} \\ p_{kj}^{(n-1)} & \text{if } d_{ij}^{(n-1)} > d_{ik}^{(n-1)} + d_{kj}^{(n-1)} \end{cases}$$

3.5 Johnson's Algorithm

Johnson's algorithm [51] was published by Donald B. Johnson in 1977. It solves all pairs shortest paths in a sparse weighted, directed graph and allows some of the edge weights to be negative numbers, but no negative-weight cycles can exist. It works by using the Bellman-Ford algorithm to compute a transformation of the input graph that removes all negative weights, allowing Dijkstra's algorithm to be used on the transformed graph [16, 8]. This algorithm may be faster than Floyd-Warshall algorithm on sparse graphs.

According to [16, 8], the re-weighting technique is used in Johnson's algorithm based on the following four steps:

- **Step 1:** It adds a new node Q with zero weight edges from it to all other nodes.
- **Step 2:** Runs the Bellman-Ford algorithm starting from the new node Q, to find each node v the minimum weight $h(v)$ of a path from Q to v. The Bellman-Ford algorithm is used to check for negative weight cycles. If this step detects a negative cycle, the algorithm is terminated.
- **Step 3:** The edges of the original graph are re-weighted using the values computed by the Bellman-Ford algorithm: an edge from u to v, having length $w(u,v)$, is given the new length $w(u, v) + h(u)h(v)$.
- **Step 4:** Q is removed, and Dijkstra's algorithm is used to find the shortest paths from each node s to every other node in the re-weighted graph.

An example of Johnson's algorithm is shown in Figure 3.5. The first three steps of Johnson's algorithm are depicted in the illustration. In Figure 3.5(a) the illustration has two negative edges, but no negative cycles. Figure 3.5(b) shows new node Q, a shortest path tree as computed by the Bellman-Ford algorithm with Q as the starting node, and the values $h(v)$ computed at each other node as the length of the shortest path from Q to that node. Note that these values are all non-positive, because Q has a length-zero edge to each node and the shortest path can be no longer than that edge. Figure 3.5(c) shows the re-weighted graph, formed by replacing each edge weight $w(u, v)$ by $w(u, v) + h(u)h(v)$. In this re-weighted graph, all edge weights are non-negative, but the shortest path between any two nodes uses the same sequence of edges as the shortest path between the same two nodes in the original graph. The algorithm concludes by applying Dijkstra's algorithm to each of the four starting nodes in the re-weighted graph.

The time complexity of Johnson's algorithm, using Fibonacci heaps in the implementation of Dijkstra's algorithm, is $O(|V|\, 2log\, |V| + |V|\, |E|)$: the algorithm uses $O(|V|\, |E|)$ time for the Bellman-Ford stage of the algorithm, and $O(|V|\, log\, |V| + |E|)$ for each of $|V|$ instantiations of Dijkstra's algorithm. Thus, when the graph is sparse, the total time can be faster than the Floyd-Warshall algorithm, which solves the same problem in time $O(|V|^3)$ [16].

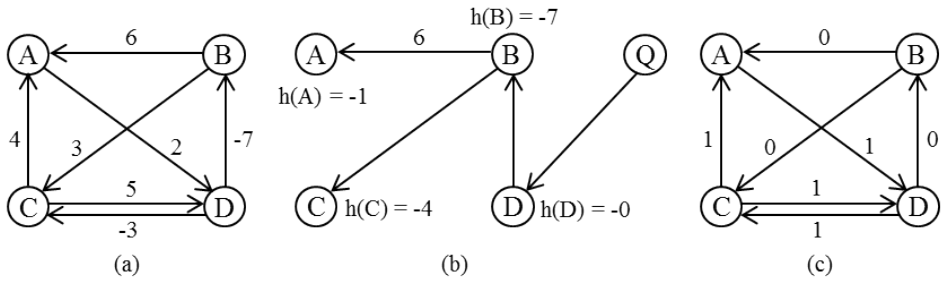

Figure 3.5: An example of Johnson's algorithm

3.6 A^*-Search Algorithm

So far we have examined search techniques that can be generalised for any network (as long as it does not contain negative length cycles); however, the physical nature of real road networks motivates the investigation of the use of heuristic solutions that exploit the near-Euclidean network structure to reduce solution times while hopefully obtaining near optimal paths. For most of these heuristics the goal is to bias a more focused search towards the destination. The A^*-Search algorithm by Hart and Nilsson [45] formalised the concept of integrating a heuristic into a search procedure. Instead of choosing the next node to label permanently as that with the least cost (as measured from the start node), the choice of node is based on the cost from the start node plus an estimate of proximity to the destination (a heuristic estimate). To build a shortest path from the origin s to the destination t, using the original distance from s accumulated along the edges (as in Dijkstra's algorithm) plus an estimate of the distance to t. Thus, consider global information about the network to guide the search for the shortest path from s to t. This algorithm places more importance on paths leading towards t than paths moving away from t.

In summary the A^*-Search algorithm combines two types of information:

- The current knowledge available on the upper bounds (given by the distance labels $d(i)$), and
- An estimate of the distance from a leaf node of the search tree to the destination.

There are several ways to estimate the lower bound from a leaf node in the search tree to the destination node. These estimations are carried out by "evaluation" functions. The closer this estimate is to a tight lower bound on the distance to the destination, the

better the quality of the A^*-Search; hence, the merits of an A^*-Search depends highly on the evaluation function $h(i, j)$. There are two main evaluation functions used in the A^*-Search. A true lower bound between two points is the length of a straight line between the two points (i.e. the Euclidean distance):

$$h_E(i,t) = \sqrt{(x(i) - x(t))^2 + (y(i) - y(t))^2}$$

where $x(i)$, $y(i)$ and $x(t)$, $y(t)$ are the coordinates for the start node i and the destination node t respectively. The other commonly used evaluation function is the Manhattan distance h_M. In this case the estimated lower bound distance is the sum of distances in the x and y coordinates.

$$h_M(i,t) = |x(i) - x(t)| + y(i) - y(t)$$

The Manhattan distance is not the true lower bound between two points, hence, will typically produce non-optimal results. The time complexity of A^*-Search depends on the heuristic. In the worst case scenario, the number of nodes expanded is exponential in the length of the solution (the shortest path), but it is polynomial when the search space is a tree, there is a single goal state, and the heuristic function h meets the following condition: $|h(x) - h^*(x)| = O(logh^*(x))$ where h^* is the optimal heuristic, the exact cost to get from x to the goal. In other words, the error of h will not grow faster than the logarithm of the "perfect heuristic" h^* that returns the true distance from x to the goal [73, 77].

3.7 Discussion

In summary, we have shown five important shortest path computation algorithms in this chapter. Our main intuition was to find the suitable shortest path computation algorithm in the context of our proposed work that can efficiently compute shortest path between two locations in the road. We choose Dijkstra's algorithm because it is more compatible with our work. Bellman-Ford algorithm is more suitable for directed graphs and it also works for undirected graph. In addition, it can work with negative weight and detect negative cycle in the graph. However, it is slower than Dijkstra's algorithm and our proposed work is based on undirected graph with non-negative weight and cycle. Floyd-Warshall algorithm works on directed graph with positive or negative edge weights but

with no negative cycle. So, the characteristics of the Floyd-Warshall algorithm do not match with the problem considered in our work. Johnson's algorithm computes all pairs shortest paths in a sparse weighted, directed graph and allows some of the edge weights to be negative numbers, but no negative-weight cycles may exist. But, We consider single source shortest paths computation in our proposed work. A^*-Search algorithm works based on heuristic solutions that exploit the near-Euclidean network structure to reduce solution times while hopefully obtaining near optimal paths. It also works for any types of network except the network with negative cycle. We have incorporated the idea of A^*-Search algorithm while shortest paths in a network are not identical. Our analysis result of the above shortest paths computation algorithms is shown in Table 3.1.

Table 3.1: Analysis of different shortest path algorithms

Algorithms	**Single Source**	**All Source**	**Negative Edge**	**Time Complexity**
Dijkstra	Y			$O(\|E\| + \|V\| \, Log \, \|V\|)$
Bellman-Ford	Y		Y	$O(\|V\| \, \|E\|)$
Floyd-Warshall		Y	Y	$O(\|V\|^3)$
Johnson		Y	Y	$O(\|V\|^2 \, Log \, \|V\| + \|V\| \, \|E\|)$
A^*-Search	Y			Depends on the heuristic

3.8 Summary

In this chapter, we have discussed the importance of finding the shortest paths for people finding their destination by road. We have reviewed shortest path algorithms and analysed the characteristics of five different shortest path algorithms, summarised the results in Table 3.1. Based on the in-depth analysis of the shortest path algorithms, we conclude that the conventional Dijkstra's algorithm is more suitable for computing shortest paths in our network; however, it is not as efficient when the shortest paths in a network are not unique. To overcome the problem of Dijkstra's algorithm we utilize the concept of A^*-Search algorithm.

Chapter 4

Proposed Methodology

Computing the shortest path between two given locations in a road network is an important problem. In the previous chapter, we discussed some popular shortest path computation algorithms. In this chapter, we investigate the exchange of raw trajectory data and its relation to privacy concerning an individual's location trajectory. For example, companies providing navigation instructions accumulate large amounts of trajectory data, which they could sell; however, it is possible that they can not sell these data directly. They are therefore looking for ways to exchange data location information with businesses across the globe. One method for them to generate revenue from these data is to sell driving instructions rather than the location information of a user.

Navigation applications are one of the most popular LBS applications, but people have to reveal their trajectory information to the service provider; however, such applications may endanger users' privacy by revealing their location and trajectory information. We observe that trajectory information consists of individual location information along with relevant trajectory facts such as navigation directions.The location information of a user in their trajectory data is removed to preserve privacy; however, other information, such as their navigation instructions, still remains within the raw trajectory data. In this thesis, we look at one particular information that is generated with individual trajectory data: driving instructions. Overall, we investigate the extent to which exchanging users' raw trajectory information preserves privacy.

The rest of this chapter is organised as follows: In Section 4.1, we present the motivation for proposing a new method. Section 4.2 includes the basic idea of how to represent road networks in our approach. Section 4.3 covers how we can generate turn instructions

for a given trajectory. In Section 4.4, we determine that sets of turn instructions are unique and thus could be used to identify a person's trajectory. Finally, in Section 4.5, we introduce a privacy level measurement based on the number of identical trajectories found for a set of given turn instructions provided by a user.

4.1 Motivation

In previous chapters, we explained that a trajectory consists of GPS coordinates (latitude and longitude) associates with the time period. Many studies [59, 60] have used Map-matching method to convert a sequence of raw GPS data into a sequence of edges in the road. Thus, our work is not based on GPS data but on sequences of edges. We utilise the sequence of edges in the graph as input in our work.

People create a vast amount of trajectory data everyday while they travel on the road. Trajectory data discloses not only positional information of the user but also a high level of detail regarding their lifestyle, preferences and habits. For example, if the LBSs service providers collate an individual's everyday data for a long period of time, they can infer detailed knowledge about the user; therefore, it is highly susceptible to privacy concerns. In this context, extensive research [2, 4, 10, 22, 34, 35, 36, 40, 50, 68] has been conducted to develop privacy preserving techniques to hide users' trajectory data. A range of research groups [2, 4, 10, 22, 34, 35, 36, 40, 50, 68] have considered these privacy issues and explored the option of suppressing trajectory information from the data that users reveal while they are using LBSs; however, trajectory data sets contain some location information at the end. For example, driving advice relevant to the specific trajectory can include turn instructions given to a user for navigation purposes.

Let us assume that, in the road network in Figure 4.1, a user, Bob wishes to go to a *Hospital* from his current position *S*. He may ask the LBS to provide a route to the *Hospital* that follows straight, straight, left and left turn directions sequentially (red arrow line). As we can see, even when removing the location points, just by using the city map and such turn instructions, we reveal that Bob commenced from S and finished at the nominated destination because this sequence is distinct for this city. Let us consider another situation in the Figure 4.1: Bob requests the path from the same start point, *S*, to the same destination, the *Hospital*, that follows straight and left turn sequences

(red dashed arrow line). Two alternative paths (black dashed arrow lines), in addition to Bob's actual path (red dashed arrow line), exist inside the map that follow the same turn instructions as Bob's path. However, it may be possible to infer the user's actual trajectory even there are though multiple paths with the same turn instructions. For example, it is possible to deduce that the actual path is the only one that ends at the point of interest, i.e., the *Hospital*. For example, if Bob visits a cancer hospital, one might infer that Bob has cancer. These two examples depict that it is possible to obtain location information without using the positional information of a user.

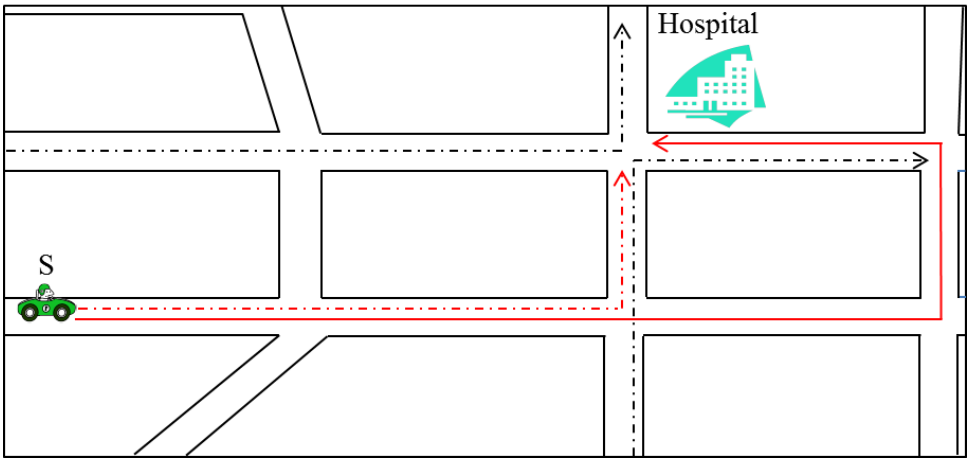

Figure 4.1: Turn directions of a user reveal the user's whereabouts

4.2 Road Network

A road network R_N can be represented as an undirected graph $G_{R_N} = (V_G, E_G)$, where V_G is a set of nodes representing intersections and E_G is a set of edges representing roads i.e., $E_G \subseteq V_G \times V_G$ consists of an edge $e_{i,j} = (n_i, n_j)$ where n_i, n_j are adjacent junctions on a road. A function $w : E_G \longrightarrow \mathbb{R}$ is used to associate a weight with each edge e. This weight can be used to model proximity between nodes, including Euclidean distance or travel time. We use $d_{G_{R_N}}(n)$ to denote the degree of a node n with respect to the graph G_{R_N}. A *route* r_p is a path on G_{R_N}, i.e., a sequence of nodes from V_G. The *length* l of a *route* r_p represents the total distance of this *route*. A *shortest path* S_p is a path that minimizes the total length that needs to be covered from a given source to a destination.

4.3 Computing Turn Instructions in a Road Network

In the previous section, we discussed the concept of road network formulation based on graph data structure. In this section, we discuss how to generate turn instructions in a road network environment. Therefore, generating turn instructions within a road network is a preliminary step in our proposed methodology. We first describe how to compute turn instructions for users' trajectories or paths from the network. Then we explain all the possible steps of the turn instructions generation algorithm in detail.

4.3.1 Generating Turn Instructions

In this section, we start by introducing the concept of turn instructions and how we compute these instructions in the road network environment. Fundamentally, turn instructions indicate a person's actions while they travel on a road; for example, left turn, right turn or continue straight. Based on the definition in [19, 79, 84], turn instructions are instructions on how to follow a route; they are task-oriented specifications of the actions to be carried out to reach a specific destination. In our work we consider three basic types of turn instructions without loss of granularity, namely left (δ_l), straight (δ_s) and right (δ_r) (see in Figure 4.2). The turn concept at road intersection points and conceptualize actions, is simply the direction to take at an intersections as opposed to viewing the road network structure.

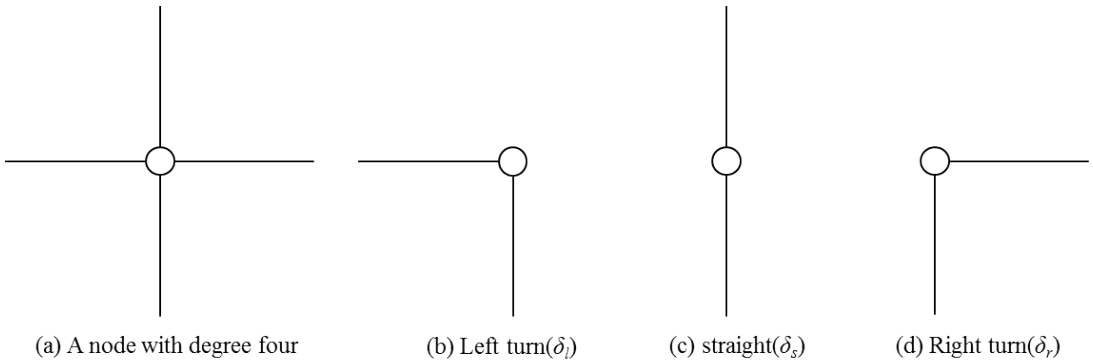

Figure 4.2: Basic turn instructions of a road network

A small set of trajectory finding turn instructions suffices to characterise most paths, path directions and path finding actions in the road. These three distinct turn instructions are considered to design our road network architecture for two reasons. Firstly, the

average degree of node ($d_{G_{R_N}}(n)$) in the road network is not more than four, conceptually (as we discussed in previous section), so the possible number of upcoming way directions can not be more than three. For example, in Figure 4.3(a), Alice wishes to go to the supermarket from her current location. The junction or intersection point is n_1 and n_1 consists with four routes, including Alice's route. Alice can reach her destination by following any route from the three possible routes: i.e., left (δ_l), straight (δ_s) and right (δ_r), respectively. Secondly, we have chosen minimal number of route finding turn instructions in our model to preserve the users' privacy, which is the most significant issue in our work (described in detail later in this subsection). Consider the following scenario in the Figure 4.3(b): n_3 is an intersection node with six edges, $\overline{n_3n_1}$, $\overline{n_3n_2}$, $\overline{n_3n_6}$, $\overline{n_3n_7}$, $\overline{n_3n_8}$ and $\overline{n_3n_4}$, respectively. We assume that $\overline{n_3n_6}$ and $\overline{n_3n_8}$ are aligned with edge $\overline{n_3n_7}$ and therefore we remove these from the network. We could reach location n_6 or even n_8 without having these edges in the network. For this, we will be able to find out more paths with the same turning patterns in the network, which could achieve more privacy for the traveller in a real road network scenario.

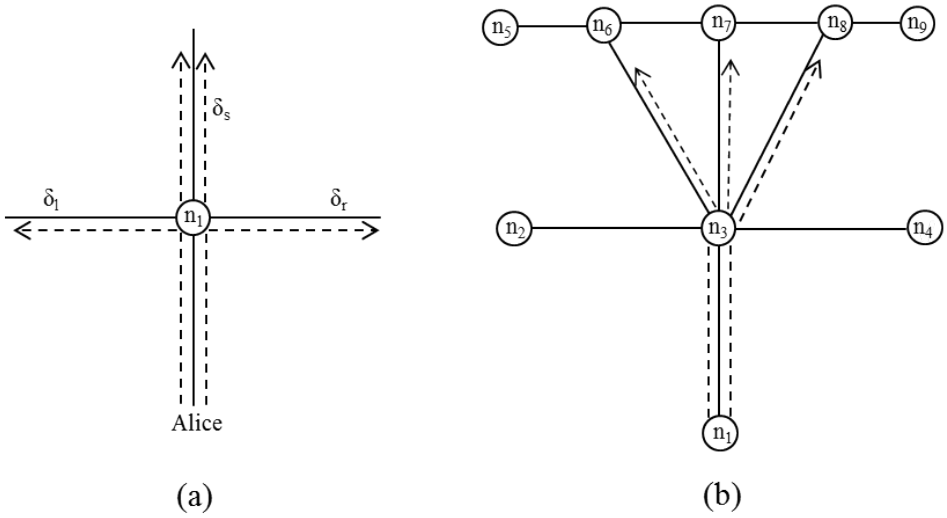

Figure 4.3: (a)Three possible ways of path finding at decision (intersection) point and (b) Set up decision point

In the following, we demonstrate how to figure out the exact turn directions at intersection points and the basic design of the intersection point in the real road network environment.

- **Determining Turn Instructions:** In a route direction context, however, people

generally do not conceptualize possible directions as vectors that originate from the centre of a decision point, i.e., the point that corresponds to the location where the branches of the intersection meet. Rather, one route segment, the one that a navigator is on, is picked out as the reference direction. On entering the intersection, the reference direction branch is combined with the subsequent branch, namely the branch that corresponds with the direction to be chosen next (see Fig 4.4). Three directions are available for conceptualizing objective oriented actions (i.e. turning or going straight). The reference direction perspective depicts the functionally relevant parts of the spatial configuration. In total, turn instructions (T_i) are extracted from the three direction model based on the reference direction perspective. Three potential directions conceptualized in combination with the reference direction result in three path finding primitives: δ_l, δ_s and δ_r (see Figure 4.2). We set the range for each reference direction between two lines to compute the turn instruction. We divide each direction into two reference angles, they are α (right angle) and β (left angle) (see in Table 4.1), we also set the range $\in$ for them.

Table 4.1: Turn instructions with their reference angles

Turning Instruction(T_i)	Reference Angles
δ_l	α_l,β_l
δ_s	α_s,β_s
δ_r	α_r,β_r

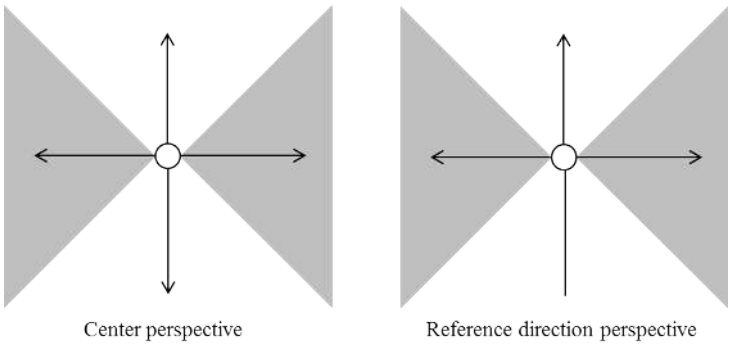

Figure 4.4: Three potential directions for a path finding context

We assume that the intersection points of G_{R_N} are aligned into the two dimensional

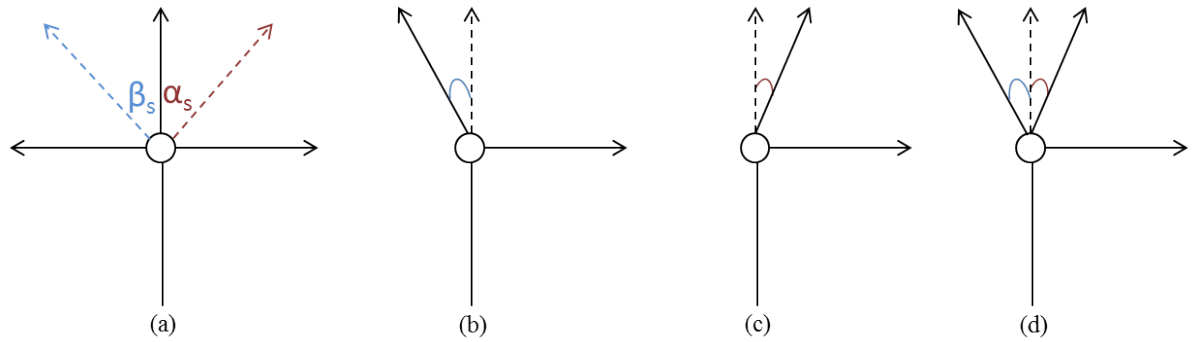

Figure 4.5: An example of forming straight instruction

space based on the Cartesian coordinate system (also called a rectangular coordinate system): horizontal and vertical lines refer to the X and Y axis respectively. An ideal "straight" instruction is established with reference angles α_s and β_s in Figure 4.5(a), if any line or edge lying in this range (called $\in$) is considered to be straight. For example, an intersection node (circle) contains three edges or lines; the left hand line fluctuates slightly left from our base perpendicular line (dashed line), but it lies in the range of α_s and it is assumed to be δ_s, as it fulfils the requirement of $\in$ (Figure 4.5(b)). In Figure 4.5(c), the right hand line is considered to be δ_s even though it shifts right from the baseline due to following the reference angle range. Another situation could occur at the intersection point due to having multiple edges with the same turn instructions. Consider the example in Figure 4.5(d): an intersection point forms four edges, and two of them are aligned into δ_s. In this situation, we assume that these two lines are identical. Based on the above consideration, we set the left and right turn instructions.

- **Removing redundant edges:** In our conceptual network model, the average degree of a node is not more than four, i.e., $\text{AVG}(d_{G_{R_N}}(n)) \leq 4$. We use the most popular graph search algorithm, i.e., the Depth-first search (DFS) algorithm, to traverse the graph data structure. DFS starts at the root and explores all nodes along each branch before backtracking. DFS also checks the condition (i.e., what we set for $d_{G_{R_N}}(n)$) for each node while exploring the G_{R_N} and removes any redundant edge if any node belongs to more than the designated degree. We describe one of the conditions for amending the intersection point in Figure 4.3. We also considered a more realistic example while designing our road network. Suppose we

have an intersection point such as is shown in Figure 4.6(a), which contains five edges (except the dashed line) i.e., $d_{G_{R_N}}(n) > 4$; we could choose any edge from this intersection point randomly, delete it and check until the requirements for using the DFS algorithm are fulfilled. For example, the right-hand sided edge is selected and deleted from the junction point. Turn instructions are then computed based on associated reference directions. Another scenario can be demonstrated in our model, such as in Figure 4.6(b): the junction node corresponds with seven edges; additional edges are removed randomly and then turn directions are generated.

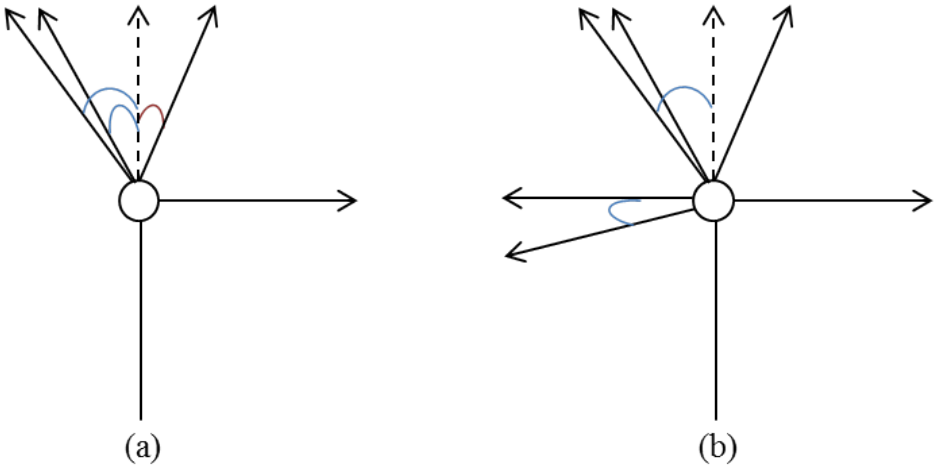

Figure 4.6: Different types of intersection in a real road network

Some intersection points in the real road network, however, can have many edges, but we assume a small number of direction instructions. As in [55, 56], a large number of turn instructions can be chosen to model the road in the real road environment. These studies [55, 56] considered sharp right, right, half right, straight, half left, left, sharp left slightly left and the like as path finding instructions (refer to Figure 4.7). This large number of path finding instructions could be beneficial in terms of having a more accurate path, but this large number of path finding instructions would cause further issues with privacy. In this thesis, our main objective is that turn instructions provided by the service provider could lead to a user's trajectory privacy violation. From this point of view, we could not consider using a large number of path finding instructions.

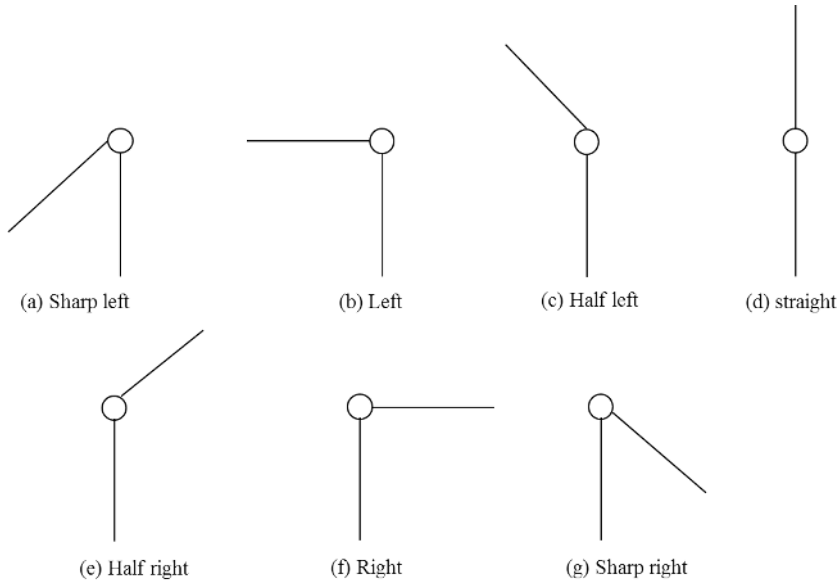

Figure 4.7: Seven conceptual path finding instructions [56]

4.3.2 Remarks on directionality

We note that once we have ("Left", "Right") or ("Left", "Straight", "Right"), such turning sequences can be travelled in both ways (except for one-way streets of course), whereas, for example, if we have ("Left", "Left"), ("Right", "Right") or even number of "Left" and "Right" turn instructions combination, such turning sequences can usually not be travelled in both directions. Consider the example in Figure 4.8, there are two different paths with different turning sequences. The path in Figure 4.8(a) follows "Right" and "Left" turn directions that can be travelled in both directions (black dashed and solid arrow lines). However, the path in Figure 4.8(b) that follows "Left" and "Left" turn directions which can not be mapped to "Right" and "Right" directions as the opposite direction is "Right" and "Right".

Generally, we advise people if they take longer sequence of turn instructions or trip with the multiple intermediate destinations/states that should not be published as a whole but only the subsequences because that will lead to a higher degree of a privacy for the user location privacy.

4.3.3 Algorithm for Generating Turn Instructions

To derive turn instructions in a road network, we propose an algorithm for generating turn instructions based on the concept above. Given a source, a destination and the

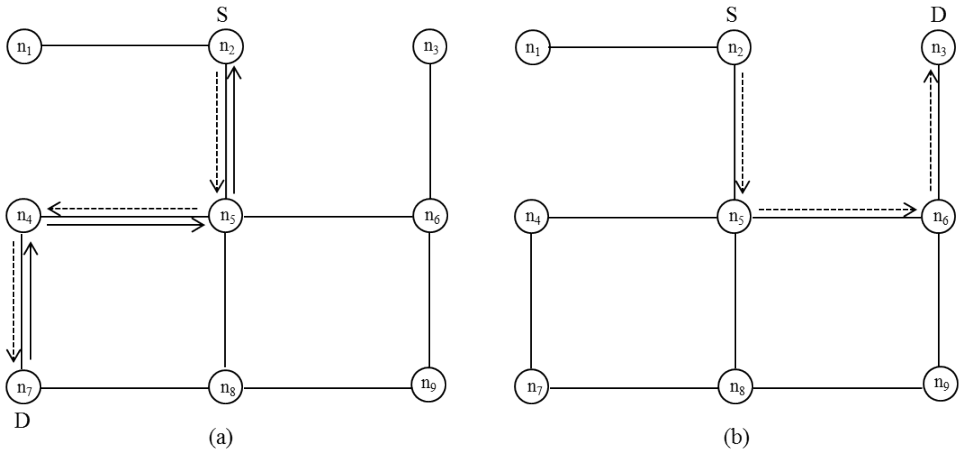

Figure 4.8: Impacts on directionality

initial orientation of a navigator, a sequence can be trivially generated using the shortest path algorithm and enumerated (the turn instructions per junction) with the above reference angle range concept. In this section, we describe the turning instructions generation algorithm step-by-step. First of all, it starts from the start node of any trip and follows the sequence of points and computes edges from that list. It then calculates the length of each edge and returns a new path (Algorithm 1 line 2). The algorithm then computes edges of the new path, calculates each edge's turning direction and the angle between them and finally checks all conditions (Algorithm 1 lines 3-6). Algorithm 1 highlights the main steps of computing turning instructions.

4.3.4 Complexity Analysis

In order to get an estimate of the computational time complexity of the generate turn instructions algorithm as presented in Algorithm 1, we begin with some notations: $|E_G|$ is the number of edges present in the graph representing the road network (read from an adjacency list); $|V_G|$ is the number of nodes. We assume the graph is not directed, so the shortest path from a to b is the same as the shortest path from b to a, although it would be simple to modify the algorithm to also deal with directed graphs. Consider line 2 of Algorithm 1 (which corresponds to the extract of minimum operations of Dijkstra's algorithm [20]), this operation would take $|E_G|$ steps as it will need to check all the edges of the graph once to compute minimum distance amongst them. Let us assume $|V_G| = n$ and so, $|E_G| = n^2$ as we know $|E_G| = |V_G|^2$ for a fully connected graph [76]. Similarly, the For

Algorithm 1: GENERATETURNINSTRUCTIONS: Generate a sequence of turn instructions

Input: Starting point n_0, destination point n_1 and a road network graph $G_{R_N} = (V_G, E_G)$

Output: A sequence of turn instructions TS

```
1 TS ← ∅
2 r_p ← shortestPath(n_0, n_1, G_{R_N})
3 for i ← 0 to |r_p| − 1 do
4     n_0 ← r_i, n_1 ← r_{i+1}
5     δ ← calculateTS(n_0, n_1)
6     TS ← TS ∪ δ
7 return TS
```

loop starting at line 3, would in a worst case in a fully connected graph, need to compute the turn instruction for every consecutive two edges of the selected shortest path leads to $(4n+1)$ steps. An overall number of steps of this algorithm are $(n^2+4n+3) \approx O(n^2)$ that leads to a computational complexity of $O(|V_G|^2)$ (the same computational complexity as the conventional Dijkstra's algorithm [20], where no computation of turn instruction is used).

One way of improving the performance of the generate turn instructions algorithm is to use a binary heap for the operation in line 2, which would lead to $O(log\,|V_G|^2)$ steps instead of $O(|V_G|^2)$.

4.4 Possible Interpretation of Navigation Instructions on actual road network

In the previous section, we discussed how to set up turning instructions and the computing of their associated instructions. In this section, we illustrate how to interpret turn instructions on a real road network, i.e, we discuss the algorithm for translating turn instructions into possible paths for the user during travel on the road.

4.4.1 Matching Turning Instructions

At first, we generate all possible shortest paths by employing the Dijkstra algorithm offline (as discussed in the previous section), and then we evaluate the turning instructions of all shortest paths from the shortest path library by using Algorithm 1. Any combination of turning instructions can be made from our turning instructions set (i.e., δ_l, δ_s and δ_r) from any location in the road network. Suppose we have a set of turn instructions T_s: a turn instruction is represented as a tuple comprising turn instructions t_p (left, right, straight) and a distance measure representing the distance travelled since the last junction d_t; thus, T_s contains a set of tuples (t_p, d_t). This is the most common method that is available in online and in-car navigation systems.

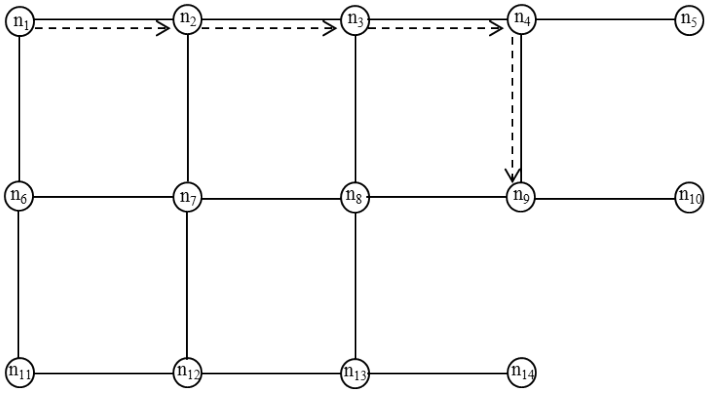

Figure 4.9: A scenario of choosing path by using the FMP algorithm

For each possible source in a road network, we first traverse each connected edge and derive the turn instruction and distance. Once derived, provided it is a match to the first element of the set, we then traverse the connected edges of the corresponding edge and look for a match to the second element in the set. This process is repeated until the number of junctions followed is equal to the length of the given turn instructions set. Each time the traversal leads to a complete match, an identical path counter is incremented. The number of resulting identical paths indicates the privacy level of a user's travelled route as the user could have travelled on any one of them.

If only one candidate path is found, the route a targeted individual used is discovered. If only a small number of paths are found, it is very likely that an adversary will be able to infer the correct path by using additional background knowledge. This problem appears to be infeasible if the given set of turn instructions is applied to a large city, we imagine that there would be many matches for any given trip when it is follow as stripped down

to its turn instructions, especially in grid-like downtown areas.

Take Figure 4.9 as an example. The "FindMatchingPaths" algorithm (FMP algorithm) finds start node n_1 and then checks adjacent nodes n_2, n_6 and checks their adjacent nodes n_3, n_7 and n_7, $n_1 1$ respectively. It is for checking the direction of every two segments in a graph. If the calculated direction matches the given direction, the algorithm adds those nodes in a path, otherwise it discards it from the path. Here, (n_1, n_2, n_3) and $(n_1, n_6, n_1 1)$ match with the first instruction from the turning sequence list so, the FMP algorithm adds (n_1, n_2, n_3) and $(n_1, n_6, n_1 1)$ to the path. Then it checks for the next instruction that matches the (n_1, n_2, n_3, n_4) and appends it to the path. Thus checking continuous until the last match is found for the instruction. Finally, it returns the nodes $(n_1, n_2, n_3, n_4, n_9)$ as the path and gives this path to the user. Algorithm 2 describes the FMP algorithm. The most efficient part of this algorithm is that it can predict the route for the user without using their exact location information. Since, the FMP algorithm only requires a set of turn instructions for getting a number of suitable routes in the road network. However, a set of turn instructions rely on positional information of a user (i.e., start and end location points) to compute turn instructions of the specific route. But, the number of resulting equivalence routes indicates the privacy level of a user's travelled route as the user could have travelled on any one of them.

Algorithm 2 describes our approach to find matching paths given a sequence of turn instructions.

4.4.2 Computational complexity

The computational complexity of the "FindMatchingPaths" algorithm (FMP algorithm) is the same as that for the generate turn instructions algorithm as discussed in Section 4.3.2. However, it is slightly faster than Algorithm 1, as it would only need to match the specific path/paths in the network for a given set of turn instructions. Consider the For loops staring at line 4 and line 6 in the Algorithm 2, would in a worst case, need to match the equivalent turn instructions based paths for a given turn instructions based path that leads to $(n + 1)$ steps and $n(n + 1)$ steps, respectively. An overall number of steps of this algorithm are $(2n^2 + 5n + 4) \approx O(n^2)$ that leads to a computational complexity of $O(|V_G|^2)$.

Algorithm 2: FINDMATCHINGPATHS: Returns the matching candidate paths given a set of turn instructions

Input: A sequence $TS = \{(tp_1, dt_1), (tp_2, dt_2), \ldots\}$, a road network graph $G_{R_N} = (V_G, E_G)$ and a counter c

Output: A list of paths with matching turn instructions to TS

```
1  P ← ∅
2  if c > |TS| then
3      return P
4  for all verticies v_0 ∈ G_{R_N} do
5      C ← connectedEdges(v_0, G_{R_N})
6      for all connected vertices (v_0, v_1) ∈ C do
7          δ_0 ← TS_c, δ_1 ← calculateTS(v_0, v_1)
8          if δ_0 ≡ δ_1 then
9              P ← P ∪ FindMatchingPaths(TS, G_{R_N}, c+1)
10 return P
```

4.5 Proposed Privacy Measurement

In summary, we have proposed a new method that can reconstruct a trajectory for a given set of turn instructions provided by a user. In this section, we introduce a privacy level that measures the uniqueness of a trajectory for a given set of turn instructions provided by a user. Let us assume G_{R_N} represents a road network, T_i denotes a set of turn instructions i.e., T_i = {'straight', 'straight', 'right'} or any combination of turn instructions, P is the path with T_i given by a user and P_l is the privacy level. The P_l is defined as follows.

Definition 2. *Let P_1 and P_2 are two paths in G_{R_N} and there turn instructions are T_{i_1} and T_{i_2}, respectively. P_1 and P_2 are turn instruction equivalent if and only if $(T_{i_1} == T_{i_2})$. The P_l of a given path P is the number of turn instruction equivalent paths.*

The privacy level is determined based on the number of equivalent turn instructions paths, i.e., the higher the number of equivalent turn instructions paths, the higher the privacy for a user. For example, consider the path of a user that has left, left, right,

straight and straight turn instructions. A service provider would like to exchange this path directions with a third party for the revenue. At first, the service provider may examine the number of equivalent turn instructions paths for the above path to see if the location of the user would be compromised if these path directions were exchanging with others. Let us assume there are 10 alternative paths, i.e., 10 equivalent turn instructions paths exist in the network for the given set of turn instructions. That means that the service provider can exchange this trip data without any worry.

4.6 Summary

We have proposed a novel approach that pinpointed without sharing trajectory location information of a user by turn instructions privacy can be breached. At first, we considered known road networks and then computed turning instructions based on turning primitives for given trajectories. We argued that the efficiency of choosing a small number of turn instructions is improved in comparison to using a larger number of turn instructions in our approach. We also described how to generate paths based on turning instructions in a real road network environment. We validate our theoretical approach through extensive experiments, which are presented in the next chapter.

Chapter 5

Experiments

In the previous chapter, we developed a technique that takes turn instructions and converts them into a trajectory. In this chapter, we will evaluate the efficacy of our proposed approach by considering for a given path, i.e., a set of turn instructions, how unique this trajectory is. The higher the number of trajectories there are for a given set of turn instructions in the road network, the more private it is considered to be. Important parameters here are node degree, trip length, trip types and grid area size. We will evaluate, in different sets of experiments, to what extent location privacy is affected. We use graph to represent road network, where nodes and edges are considered as intersections and roads between two intersections (see Section 4.2). We use both synthetic and real data sets in our experimental analysis.

The remainder part of this chapter is organized as follows: in Section 5.1, we present an extensive experimental evaluation of our proposed approach based on synthetic data sets, to evaluate the effectiveness by considering different parameters. We measure the privacy level based on the number of paths are found for a given set of turn instructions by a user. Then in Section 5.2, we evaluate the efficiency of our proposed algorithm by considering real data sets to measure the above mentioned privacy level.

5.1 Experimental Evaluation based on Simulated Data Sets

In this section, we discuss the setting of the experiment's environment for evaluation based on our proposed work efficiency. Furthermore, we define a set of parameters to measure

the privacy level (as discussed in Chapter 4), which affects the algorithm's accuracy.

5.1.1 Experimental Setup

First, we evaluate our algorithm for synthetic data sets. We consider a 20×30 grid that consists of 600 nodes, with an average node degree of four. We also consider average node degrees of three and two by pruning some edges from the network. We randomly generate 100 of the shortest paths by using Dijkstra's algorithm in our graph area and apply the methods to obtain the turn instructions between a source and destination and then reverse engineer them.

We define the parameters we have considered for evaluating our approach based on synthetic data sets. Node degree, trip length, trip types and grid area size are the parameters associated to the experiments. Table 5.1 summarizes the values used for each parameter in our experiments and their default values.

- **Node degree**

  We assume a graph to represent a road network that consists of a set of edges and a set of nodes. Node degree refers to the number of edges that belong to a node. For example, intersection of a road may consist of 2 or more road links.

- **Trip length**

  We calculate turn instructions for every trip in a graph by utilizing a turn instructions generation algorithm. We define the trip length as the number of turn instructions that belong to a trip. For example, if we consider a trip that follows straight, straight and left turn instructions, then this trip has a trip length of three.

- **Trip types**

  Trip types define how many points of interest a user visited during one trip. For example, if a user goes to the kindergarten from her home to pick up her child, but she stops at point of interest, say a coffee shop, and then goes to her main destination, i.e., kindergarten, her trip is to be considered as one stop type trip.

- **Grid area size**

  Grid area size refers to the size of each grid, i.e., graph network area. For example, 3×2 grid area consists 6 nodes, 3×3 grid area consists 9 nodes, etc.

Table 5.1: Experimental setup

Parameter	Value of a Parameter	Default
Node Degree	2, 3, 4	4
Trip length	10, 20, 30, 40, 50	20
Trip Types	0 stop, 1 stop, 2 stops	0 stop
Grid area size	20×30, 30×20	20×30

Our algorithm was implemented in Python programming language. We ran the experiments on a desktop with a Pentium 3.40 GHz CPU and 8 GByte RAM. We present experimental results of our proposed algorithm based on simulated data sets in Section 5.1.2.

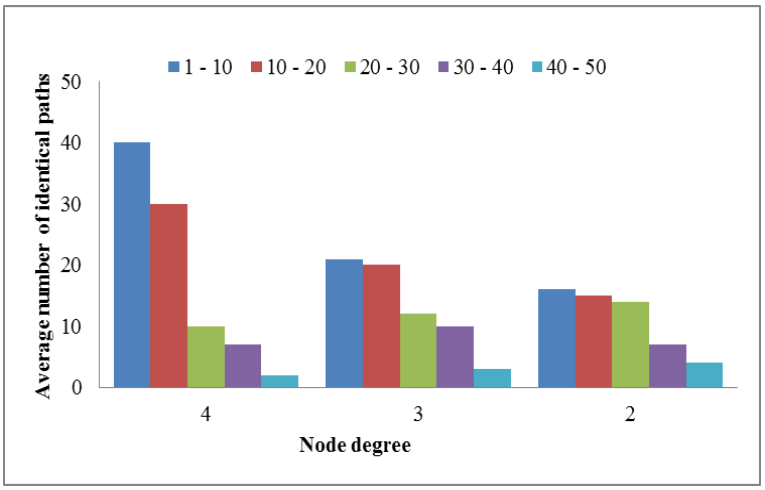

Figure 5.1: Impact in privacy level by varying node degree for a given turn instructions (1-10, 10-20, 20-30, 30-40, 40-50)

5.1.2 Experimental Results for Simulated Data

In this section, we analyse the results obtained for synthetic data based experiments. Each graph below represents the privacy level measurement reported for each experiment.

Effect of Node Degree

In this set of experiments, we vary the node degree from 2 to 4 in the network. A large number of identical paths for a given set of turn instructions represent a more precise location of the user and thus ensures a higher level of privacy.

Figure 5.1 shows the result of measuring privacy level by comparing the total number of identical trips with the average node degree 4, 3 and 2 in the graph network. We observe that the higher number of paths are found for a given set of turn instructions when the average number of node degrees is 4. This is due to the existence of more edges for each intersection when the average node degree is 4 in the graph. We see that a significantly higher number of identical paths are found for a given small set of turn instructions. For example, the given set of turn instructions $1 - 10$ in the figure. On the other hand, a low number of identical paths are returned for a given large set of turn instructions, for example, the given set of turn instructions $40 - 50$ in the figure. We also observe with a decrease of the average number of node degree the number of identical paths are found by a given set of turn instructions, is lower than that of the average number of node degree 4. Number of edges are pruned from each intersection for considering the number of average node degree criteria for validating our arguments and therefore the number of paths are significantly low. However, the number of identical paths for a given large set of turn instructions are higher than the average number of node degree 4.

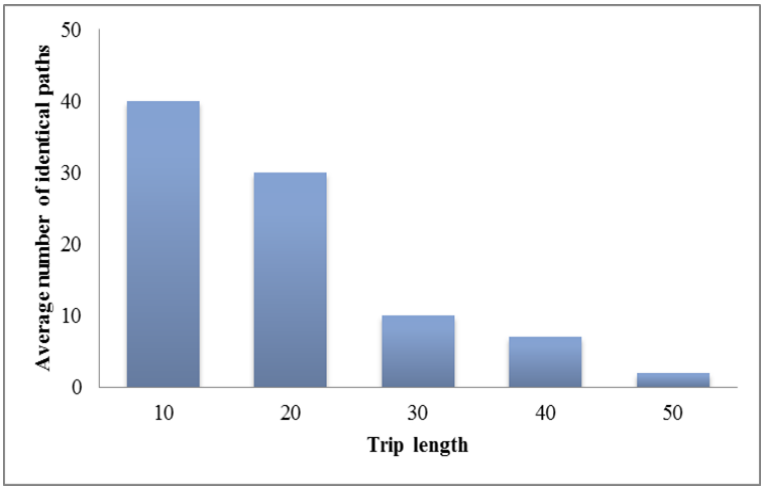

Figure 5.2: Impact in privacy level by varying trip length

Effect of Trip Length

We vary the trip length of the synthetic data sets as 10, 20, 30, 40 and 50, and observe that our proposed algorithm can efficiently find out the number of identical paths for a given set of turn instructions considering the trip length variation. We consider that the maximum number of trip length is 50, due to having difficulty finding a trip with a higher trip length; in the worst case, it is possible to find a minimum of one path for a provided set of turn instructions.

In Figure 5.2, 40 identical paths are found among 100 paths for a given set of turn instructions or trip length ranging from 1 to 10. In contrast, less than 10 identical paths are found for a given trip length of 40 to 50. That means that the longer the trip length, the lower the number of trips found in the network.

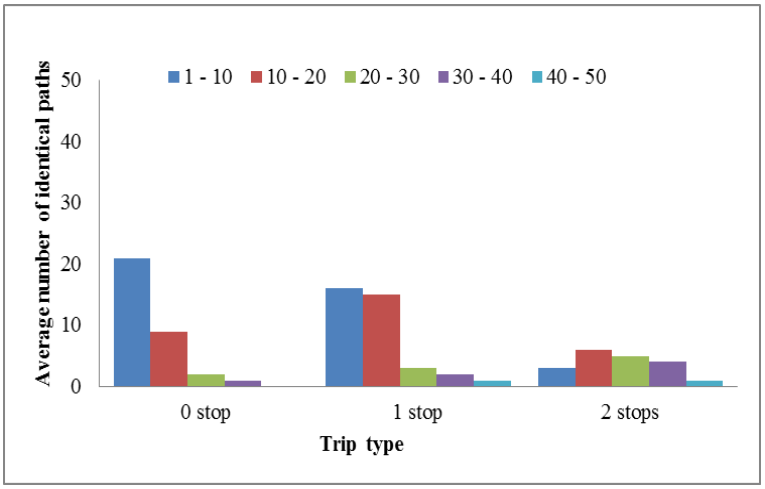

Figure 5.3: Impact in privacy level by comparing trip types with the number of identical paths are found for a given turn instructions(1-10, 10-20, 20-30, 30-40, 40-50)

Effect of Trip Types

In this set of experiments, we observe the privacy level by varying trip types when a number of identical trips are found. We consider three types of trip such as 0 stop, 1 stop and 2 stops for evaluating the performance of our proposed algorithm. Figure 5.3 delimits the result of measuring privacy level by comparing the total number of trips with the trip types of those trips in the network. We observe that the number of identical trips for a given set of turn instructions decreases with an increase in the number of stop points. For example, the number of trips are found is high and low while we consider the

0 stop trip type and the 2 stops trip type, respectively. The higher the number of trips, the higher the level of privacy in terms of trip types. We also observe that trips with a small trip length are more frequent in terms of trip types because most of the shorter trips consist with 0 stop type trips. For example, if we consider a trip with trip length 3, say straight, straight and left turn instructions, then this trip is considered as a short trip and it has less potential to contain a high number of stop points. We see in the figure that most of the paths with path length 1 - 10 are produced in 0 stop type trip. On the other hand, trips with a higher number of trip length are not as frequent in terms of trip types. We also see that the number of trips with a longer trip length is higher in 2 stop trip types compared with 0 stop trip type because it is has possible to have multiple stop points for a longer trip with a high trip length. However, a longer trip with a higher trip length returns fewer identical paths, i.e., privacy level is low, which indicates that the privacy of a user is preserved when there is a large number of trips along with the same trip instructions.

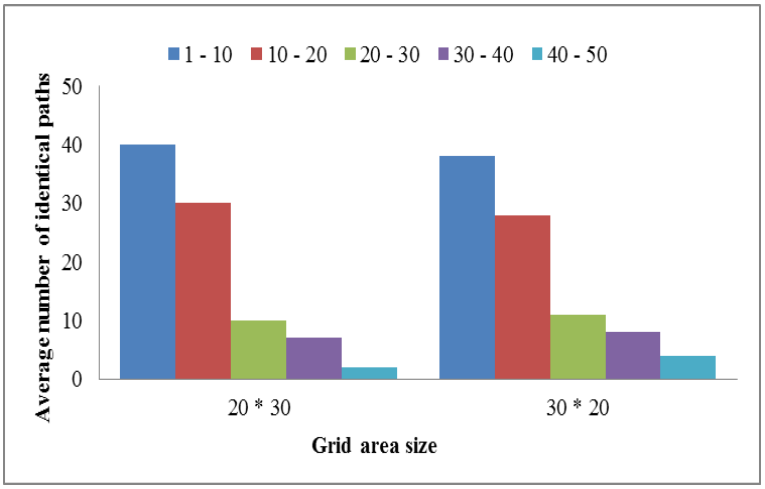

Figure 5.4: Impact in privacy level by comparing trip types with the number of identical paths are found for a given turn instructions(1-10, 10-20, 20-30, 30-40, 40-50)

Effect of Grid Area Size

In these experiments, we observe the effect of the grid area size on the level of privacy. We consider 20×30 and 30×20 grid area sizes for evaluating the performance of our proposed approach. The effect of varying grid area size is not significant in terms of getting the number of identical paths for a given set of turn instructions. However, it

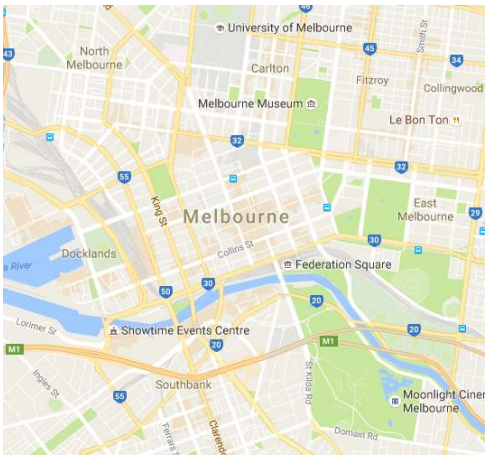

Figure 5.5: Melbourne downtown area

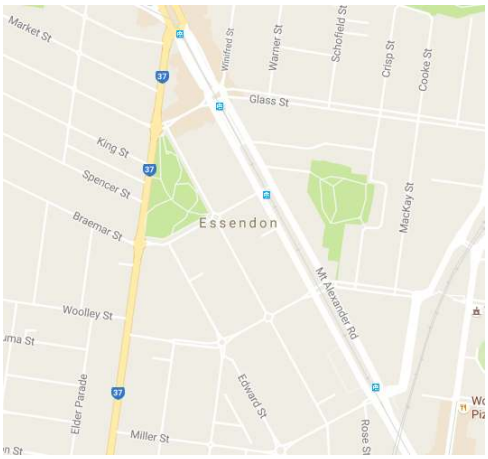

Figure 5.6: Essendon inner suburb

affects the patterns of the turn instructions, along with the path. For example, consider two small grid areas: 3×4 and 4×3. One path has the same trip length, say 3, and has the same start and destination point. The path in the former grid area is straight, right and straight, whereas this same path follows straight, straight and right in the latter one.

We observe in Figure 5.4 that the ratio of the level of privacy between 20×30 and 30×20 grid areas is nearly same for any size trip length; the higher the trip length the lower the number of identical paths are found, which indicates a lower level of privacy. In contrast, the lower the trip length, the higher number of identical trips, which indicates higher level of privacy for a user. We also observe that the number of longer trips with the longer trip length for a given set of trip instructions slightly increases in the latter grid area than the former grid area.

In summary, a significant number of trips, even in populated areas with grid like infrastructure, can be disclosed easily without the need of additional background information. We also evaluate our proposed algorithm performance, considering real road network in the next section.

5.2 Experimental Evaluation based on Real Data sets

In the previous section, we observed the performance and behaviour of our proposed method in a synthetic data sets setting. In this section, we evaluate the effectiveness of our proposed algorithm based on real-life data sets. In our evaluation of real road network data, we consider the privacy level to be based on the measurement of finding the number

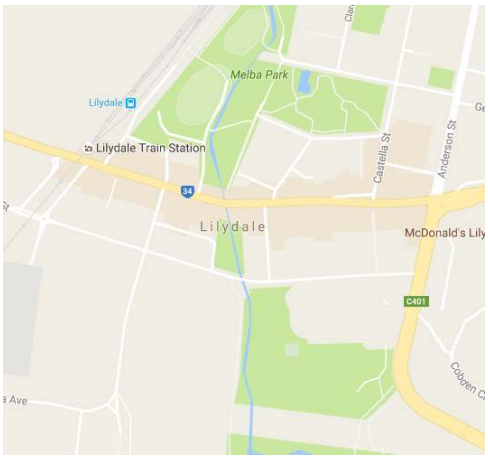

Figure 5.7: Lilydale outer suburb

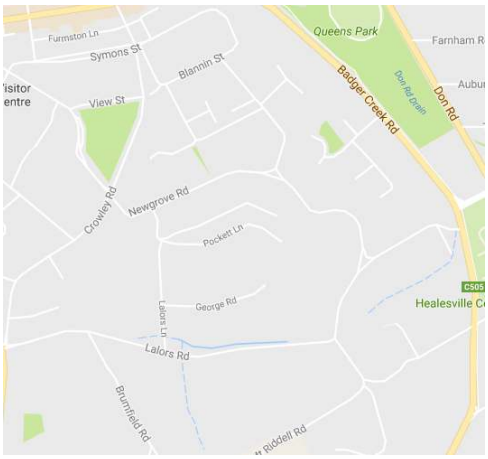

Figure 5.8: Healesville rural area

of paths with unique turn instructions of a specified path, given a road network (see our paper [47]).

We considered a real road network where we have considered Euclidean distance between two locations in the road. However, we did not consider distances in synthetic data. Because, in synthetic data setting, we just wanted to see how many simple instructions it takes to reverse engineer a path. With realistic data, if we do not consider distance between two locations then it may be possible to get a very long path for a given long set of turn instructions that is inapplicable. Let us assume, we have a set of turn instructions without considering distances, it may possible to get a very long path for example from the centre of the city to the rural area which is not acceptable to the user. We considered the city of Melbourne area for the evaluation of our algorithm and evaluated the performance of the algorithm by comparing the number of occurrences of the path can be found in the network for a given path i.e., a set of turn instructions. Our observation is that even when a seemingly large number of combinations possible, user paths are still unique on average i.e., only one path is available for the given set of turn instructions.

The main objective of our evaluation was to determine the conditions that can lead to turn instructions exposing the route of a user travelled and what circumstances allow for turn instructions to be safely exchanged. We found that there are many situations where user privacy is exposed if turn instructions are shared.

We considered the city of Melbourne, Australia. This is due to the road network in the inner city approximating a grid while in the suburbs the road network is more sparse

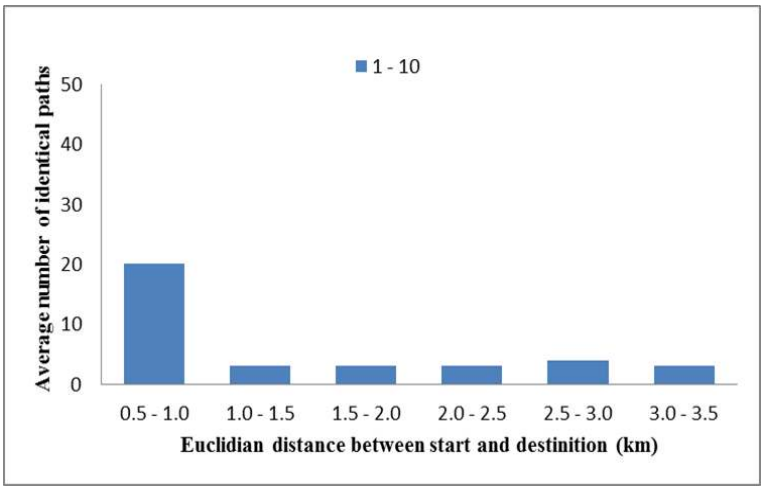

Figure 5.9: Melbourne downtown - average number of identical paths found for generated paths by the number of turn instructions (1-10, 10-20, 20-30, 30-40)

which is general in many cites around the world.

The actual trajectory of a user can be indirectly determined in instances where only a few identical paths can be identified. Moreover, the privacy of a user is preserved when a large number of paths have the same turn instructions.

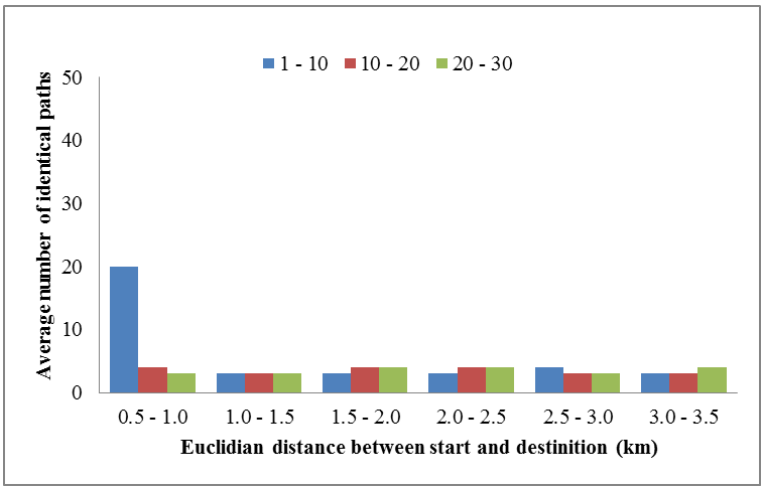

Figure 5.10: Essendon - average number of identical paths found for generated paths by the number of turn instructions (1-10, 10-20, 20-30)

Paths residing in Melbourne's inner city and surrounding were generated with turn sequences and used to search for identical sequences. The road OpenStreetMap [43]. A bounding box with an area of $4km^2$ was placed around areas of consideration. We generated 100 paths within the test areas. Paths were generated randomly selecting a source and destination pair. The shortest path is then computed using Dijkstra's

algorithm and the turn sequences are derived.

The Melbourne downtown is arranged in a grid like manner with many intersections leading into main arterials as well as smaller streets and lanes as can be seen in Figure 5.5. Our results demonstrated in Figure 5.5 that even when a apparently large number of combinations possible, user paths are still unique on average.

Such results were found in the inner city suburb of Essendon and suburbs in its extended area in Figure 5.10. Essendon is organized in a grid like structure as can be seen in Figure 5.6 and has a reasonably high population density for an inner city suburb.

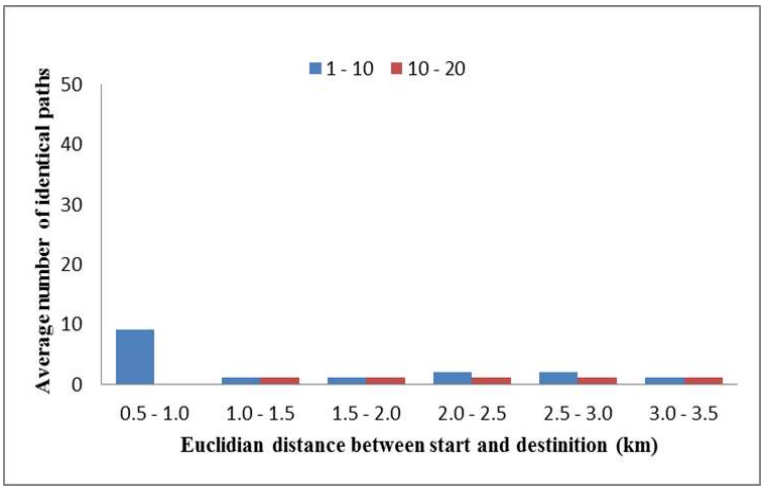

Figure 5.11: Lilydale - average number of identical paths found for generated paths by the number of turn instructions (1-10, 10-20)

In the outer suburbs, it is easier to determine a path a user takes due to the graph being less interconnected. Consider the suburb of Lilydale in Figure 5.7, it can be seen that there are sections of the road network that are not interconnected. The results charted in Figure 5.11 indicate that at most nine identical turn sequences can be derived and this reduces down to two identical sequences for longer turn instruction sequences.

Rural areas unlike city areas have simplified road networks. Consider the map of Healesville, a semi-rural town. It can be seen in Figure 5.8 that the road network for Healesville is simplified compared to the areas closer to the city. Our results given in Figure 5.12.

From experiments, we conclude that a large number of paths, even in reasonably populated areas with grid like infrastructure, can be revealed easily without the need of additional background information. Our findings indicate that data providers that

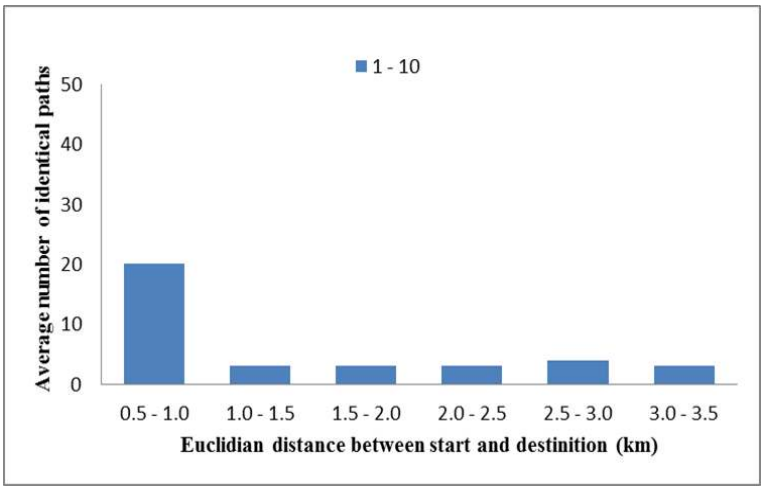

Figure 5.12: Healesville - average number of identical paths found for generated paths by the number of turn instructions (1-10, Note: paths with longer turn instructions could not satisfy the road network)

wish to exchange turn instructions information in a privacy-preserving manner need to consider adding noise using techniques such as differential privacy [26] to ensure privacy is protected.

5.3 Summary

In this chapter, we evaluated the accuracy and robustness of our algorithm based on synthetic and real data sets. Algorithm accuracy based on synthetic data was evaluated for node degree, trip length, trip types and grid area size. We observed that a high level of privacy is gained for shorter trips rather than longer trips in the network. We also evaluated our proposed method by considering real-life data sets. However, we set-up the real data sets environment slightly differently than the synthetic data set environment due to the adaptability with our method to the real road map. We observed that even without location data of a user, privacy can be breached by analysing turn instructions. We found that there are many situations where user privacy is exposed if turn instructions are shared. To the best of our knowledge, this is the first work to address the problem of keeping turn instructions with the user profile without having the location information of a user.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we have presented and systematically highlighted that even without the location data of a user, privacy can be violated by using turn instruction data.

We introduced a turn instructions based approach that shows that an individual's location can be revealed from his/her profile without using their actual location information. Removing location information from the user profile could be considered as privacy preserving; however, a number of significant aspects related to a user's position, such as navigation instructions, can still be stored in the user's profile. We demonstrate here that it is possible to reconstruct a path based on the user's navigation instructions; therefore, the user's location privacy is at risk.

We have described our approach for deriving the turn instructions between a source and destination. This approach uses a set of turn instructions, traverses every node in a road network and incrementally expands until a match, i.e., a path or paths based on the given turn instructions, can be found. We also propose a method that helps to reverse engineer the path based on the set of turn instructions.

We have tested our algorithm for both synthetic and real data. We have used synthetic data to test the algorithm's behaviour and accuracy for various trajectory parameters. The parameters evaluated were trip length, trip types, average node degree and grid area size. We have shown that by considering the above parameters, our proposed algorithm can successfully reconstruct routes for given turn instructions. We also observe that paths with fewer turn instructions are more private than longer paths with longer turn

instructions.

We used real-life data to observe the applicability of our algorithm in the real road network environment. In this experiment setting, we considered Euclidean distance between two locations in the road; however, we did not consider any distances in synthetic data. We just wanted to see how many simple instructions it takes to reverse engineer a path. Using real data, we have shown that our proposed algorithm can efficiently rebuild paths for given turn instructions. We observed that the paths with longer turn instructions are infrequent in the network.

The main intention of our evaluation was to determine the conditions that can lead to turn instructions, exposing the user's route and circumstances that allow for turn instructions to be safely exchanged. We found that there are many situations where user privacy is exposed if turn instructions are shared.

LSPs need to exchange navigation or turn instructions with partners with care, as the original path of a user can be inferred in the majority of cases despite the fact that the data is cleansed of GPS traces such as source and destination locations. We suggest that LSPs use our proposed algorithm to observe the number of outcome paths (no. of path $>$ 1) for a given set of navigation instructions before they exchange navigation instructions with third parties. On the basis of this evaluation, they may release the user's turn instructions data, which gives multiple paths and avoids disclosing turn instructions data sets that return fewer distinct paths.

6.2 Future Work

In the future, we intend to explore the following research directions in the context of preserving the user's trajectory privacy with the aim of enhancing the accuracy of navigation instructions and user location privacy.

6.2.1 Integration with a Large Number of Turn Instructions

One promising research direction is to determine the appropriate number of turn instructions and the utility of those instructions in the real road network, since the real road network consists of different types of intersection points, for example, t-intersection, 3-way intersection, 4-way intersection, roundabouts, etc. Roundabout intersections can

include more than four routes; therefore, a large number of turn instructions are required for reconstructing the route accurately. In this thesis, we have considered three basic types of turn instructions for demonstrating situations violating user's trajectory location privacy. A number of existing approaches [55, 56] use a large number of turn instructions instead of using coarse turn instructions. In the future, we plan to integrate a large number of turn instructions into our approach; therefore, we will examine the efficacy of our approach based on a large number of turn instructions. A large number of turn instructions can be chosen to model the road if it achieves a more accurate path; however, improved path accuracy could endanger the user's location privacy.

Consider the example in Figure 6.1. We assume that seven different types of turn instructions, as described in [56], manipulate this road network. A user wants to go to the *hospital* (H_1) from his current position S by requesting a path that follows straight and half right turn instructions. As we can see, there is only one path (large dashed line) that follows the designated turn instructions given by the user, but any person with a city map and the turn instructions can determine the user's position. Alternatively, the user can consider a path based on straight, straight and right turn instructions, which generates three alternative routes, i.e., route 1, route 2 and route 3. Route 1 leads to the *hospital* (H_1), i.e., the user's desired destination; route 2 leads to a *cafe*, and route 3 ends up at the *hospital* (H_2). In this scenario, it is very difficult to any third person to determine the user's actual position, though the turn instructions information of the user has already been inferred. As a result, obtaining multiple paths from turn instructions, given by a user, achieved more privacy for that user. We aim to examine the above challenges while incorporating different types of turn instructions in the real road network setting.

6.2.2 Privacy-preserving Methods

We plan to investigate privacy-preserving methods that can allow for the exchange of turn instructions safely. For example, instead of releasing the entire path, we could utilise the sub-paths and study their effect on privacy integrity. Another possible method is that, in exchange for revealing every turn instruction in a path, we could consider some turn instructions as unidentified. For example, consider a path that follows left, straight, straight and right turn instructions; instead of releasing the same turn instructions se-

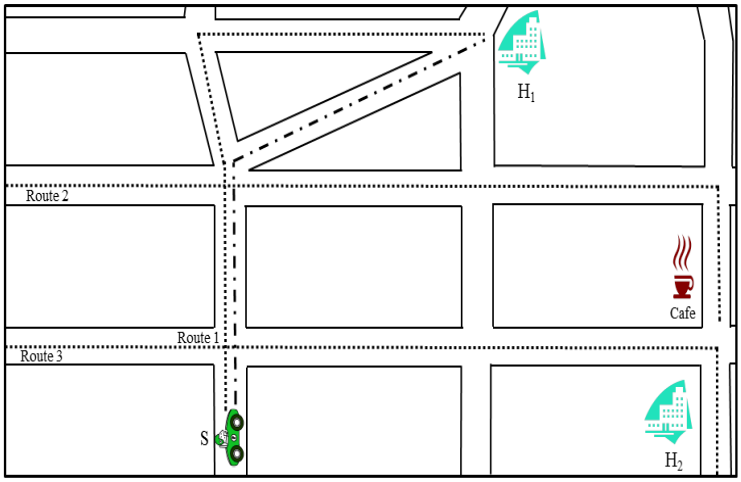

Figure 6.1: A scenario of choosing a large number of turn instructions

quentially, we release ?, straight, straight, right or left, ?, straight, right or left, straight, ?, right or left, straight, straight, ?. We aim to utilise the above mentioned methods to extend our proposed approach in future.

Bibliography

[1] Microsoft. location privacy: Where are we headed? 2011. URL `http://www.microsoft.com/privacy/dpd`.

[2] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 376–385. Ieee, 2008.

[3] Ravindra K Ahuja. *Network flows*. PhD thesis, TECHNISCHE HOCHSCHULE DARMSTADT, 1993.

[4] Bhuvan Bamba, Ling Liu, Peter Pesti, and Ting Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceedings of the 17th international conference on World Wide Web*, pages 237–246. ACM, 2008.

[5] Richard Bellman. A markovian decision process. Technical report, DTIC Document, 1957.

[6] Richard Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16: 87–90, 1958.

[7] Alastair R Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive computing*, 2(1):46–55, 2003.

[8] Paul E. Black. Johnson's algorithm. In *Dictionary of algorithms and data structures*. 2004.

[9] David Caduff and Sabine Timpf. The landmark spider: Representing landmark knowledge for wayfinding tasks. In *AAAI spring symposium: Reasoning with mental and external diagrams: Computational modeling and spatial assistance*, pages 30–35, 2005.

[10] Reynold Cheng, Yu Zhang, Elisa Bertino, and Sunil Prabhakar. Preserving user location privacy in mobile data management infrastructures. In *Privacy Enhancing Technologies*, pages 393–412. Springer, 2006.

[11] Boris V Cherkassky and Andrew V Goldberg. Negative-cycle detection algorithms. *Mathematical Programming*, 85(2):277–311, 1999.

[12] Chi-Yin Chow and Mohamed F Mokbel. Enabling private continuous queries for revealed user locations. In *International Symposium on Spatial and Temporal Databases*, pages 258–275. Springer, 2007.

[13] Chi-Yin Chow and Mohamed F Mokbel. Trajectory privacy in location-based services and data publication. *ACM Sigkdd Explorations Newsletter*, 13(1):19–29, 2011.

[14] Chi-Yin Chow and Mohemad F Mokbel. Privacy of spatial trajectories. In *Computing with spatial trajectories*, pages 109–141. Springer, 2011.

[15] Chi-Yin Chow, Mohamed F Mokbel, and Xuan Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 171–178. ACM, 2006.

[16] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001. ISBN 0070131511.

[17] Heriberto Cuayáhuitl, Nina Dethlefs, Lutz Frommberger, Kai-Florian Richter, and John Bateman. Generating adaptive route instructions using hierarchical reinforcement learning. In *International Conference on Spatial Cognition*, pages 319–334. Springer, 2010.

[18] Maria Luisa Damiani, Elisa Bertino, and Claudio Silvestri. Protecting location privacy against spatial inferences: the probe approach. In *Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS*, pages 32–41. ACM, 2009.

[19] Michel Denis, Francesca Pazzaglia, Cesare Cornoldi, and Laura Bertolo. Spatial discourse and navigation: An analysis of route directions in the city of venice. *Applied cognitive psychology*, 13(2):145–174, 1999.

[20] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[21] Matt Duckham and Lars Kulik. simplest paths: Automated route selection for navigation. In *Spatial information theory. Foundations of geographic information science*, pages 169–185. Springer, 2003.

[22] Matt Duckham and Lars Kulik. A formal model of obfuscation and negotiation for location privacy. In *Pervasive computing*, pages 152–170. Springer, 2005.

[23] Matt Duckham and Lars Kulik. Simulation of obfuscation and negotiation for location privacy. In *International Conference on Spatial Information Theory*, pages 31–48. Springer, 2005.

[24] Matt Duckham and Lars Kulik. Location privacy and location-aware computing. *Dynamic & mobile GIS: investigating change in space and time*, 3:35–51, 2006.

[25] Matt Duckham, Lars Kulik, and Michael Worboys. Imprecise navigation. *GeoInformatica*, 7(2):79–94, 2003.

[26] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[27] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876, pages 265–284. Springer, 2006.

[28] Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345–, June 1962. ISSN 0001-0782. doi: 10.1145/367766.368168. URL `http://doi.acm.org/10.1145/367766.368168`.

[29] Michael L Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3): 596–615, 1987.

[30] Julien Freudiger, Maxim Raya, Márk Félegyházi, Panos Papadimitratos, et al. Mix-zones for location privacy in vehicular networks. 2007.

[31] Benjamin Fung, Ke Wang, Rui Chen, and Philip S Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (CSUR)*, 42 (4):14, 2010.

[32] Stefan Funke and Sabine Storandt. Path shapes: an alternative method for map matching and fully autonomous self-localization. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 319–328. ACM, 2011.

[33] Stefan Funke, Robin Schirrmeister, Simon Skilevic, and Sabine Storandt. Compass-based navigation in street networks. In *International Symposium on Web and Wireless Geographical Information Systems*, pages 71–88. Springer, 2015.

[34] Bugra Gedik and Ling Liu. Protecting location privacy with personalized k-anonymity: Architecture and algorithms. *Mobile Computing, IEEE Transactions on*, 7(1):1–18, 2008.

[35] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. Mobihide: a mobilea peer-to-peer system for anonymous location-based queries. In *Advances in Spatial and Temporal Databases*, pages 221–238. Springer, 2007.

[36] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. Prive: anonymous location-based queries in distributed mobile systems. In *Proceedings of the 16th international conference on World Wide Web*, pages 371–380. ACM, 2007.

[37] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private queries in location based services: anonymizers are not necessary. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 121–132. ACM, 2008.

[38] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 246–255. ACM, 2009.

[39] Reginald G Golledge. *Wayfinding behavior: Cognitive mapping and other spatial processes*. JHU press, 1999.

[40] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[41] Marco Gruteser and Dirk Grunwald. Enhancing location privacy in wireless lan through disposable interface identifiers: a quantitative analysis. *Mobile Networks and Applications*, 10(3):315–325, 2005.

[42] Marco Gruteser and Baik Hoh. On the anonymity of periodic location samples. In *International Conference on Security in Pervasive Computing*, pages 179–192. Springer, 2005.

[43] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[44] Shazia Haque, Lars Kulik, and Alexander Klippel. Algorithms for reliable navigation and wayfinding. In *International Conference on Spatial Cognition*, pages 308–326. Springer, 2006.

[45] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[46] Amina Hossain, Al-Amin Hossain, Sung-Jae Jang, Young-Sung Shin, and Jae-Woo Chang. K-anonymous cloaking algorithm based on weighted adjacency graph for preserving location privacy. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 358–365. IEEE, 2012.

[47] Amina Hossain, Anthony Quattrone, Egemen Tanin, and Lars Kulik. On the effectiveness of removing location information from trajectory data for preserving location privacy. In *Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pages 49–54. ACM, 2016.

[48] Haibo Hu and Dik Lun Lee. Range nearest-neighbor query. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):78–91, 2006.

[49] Haibo Hu and Jianliang Xu. 2pass: Bandwidth-optimized location cloaking for anonymous location-based services. *Parallel and Distributed Systems, IEEE Transactions on*, 21(10):1458–1472, 2010.

[50] Zheng Huo, Xiaofeng Meng, Haibo Hu, and Yi Huang. You can walk alone: trajectory privacy-preserving through significant stays protection. In *Database Systems for Advanced Applications*, pages 351–366. Springer, 2012.

[51] Donald B Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM (JACM)*, 24(1):1–13, 1977.

[52] Panos Kalnis, Gabriel Ghinita, Kyriakos Mouratidis, and Dimitris Papadias. Preventing location-based identity inference in anonymous spatial queries. *Knowledge and Data Engineering, IEEE Transactions on*, 19(12):1719–1733, 2007.

[53] Ali Khoshgozaran and Cyrus Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *Advances in Spatial and Temporal Databases*, pages 239–257. Springer, 2007.

[54] Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. An anonymous communication technique using dummies for location-based services. In *Pervasive Services, 2005. ICPS'05. Proceedings. International Conference on*, pages 88–97. IEEE, 2005.

[55] Alexander Klippel and Daniel R Montello. Linguistic and nonlinguistic turn direction concepts. In *International Conference on Spatial Information Theory*, pages 354–372. Springer, 2007.

[56] Alexander Klippel, Heike Tappe, Lars Kulik, and Paul U Lee. Wayfinding choremesa language for modeling conceptual route knowledge. *Journal of Visual Languages & Computing*, 16(4):311–329, 2005.

[57] Alexander Klippel, Thora Tenbrink, and Daniel R Montello. The role of structure and function in the conceptualization of directions, 2010.

[58] Kristen LeFevre, David J DeWitt, and Raghu Ramakrishnan. Mondrian multidimensional k-anonymity. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, pages 25–25. IEEE, 2006.

[59] Hengfeng Li, Lars Kulik, and Kotagiri Ramamohanarao. Spatio-temporal trajectory simplification for inferring travel paths. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 63–72. ACM, 2014.

[60] Hengfeng Li, Lars Kulik, and Kotagiri Ramamohanarao. Robust inferences of travel paths from gps trajectories. *International Journal of Geographical Information Science*, 29(12):2194–2222, 2015.

[61] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. Closeness: A new privacy measure for data publishing. *Knowledge and Data Engineering, IEEE Transactions on*, 22(7):943–956, 2010.

[62] Bing Liu. Using knowledge to isolate search in route finding. In *IJCAI*, pages 119–125. Citeseer, 1995.

[63] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[64] Tomasz Marciniak and Michael Strube. Classification-based generation using tag. In *Natural Language Generation*, pages 100–109. Springer, 2004.

[65] Tomasz Marciniak and Michael Strube. Modeling and annotating the semantics of route directions. In *Proceedings of the 6th International Workshop on Computational Semantics*, pages 12–14, 2005.

[66] David M Mark. Automated route selection for navigation. *Aerospace and Electronic Systems Magazine, IEEE*, 1(9):2–5, 1986.

[67] Mohamed F Mokbel, Chi-Yin Chow, and Walid G Aref. The new casper: query processing for location services without compromising privacy. In *Proceedings of the 32nd international conference on Very large data bases*, pages 763–774. VLDB Endowment, 2006.

[68] Mehmet Ercan Nergiz, Maurizio Atzori, and Yucel Saygin. Towards trajectory anonymization: a generalization-based approach. In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, pages 52–61. ACM, 2008.

[69] Balaji Palanisamy and Ling Liu. Mobimix: Protecting location privacy with mix-zones over road networks. In *2011 IEEE 27th International Conference on Data Engineering*, pages 494–505. IEEE, 2011.

[70] Balaji Palanisamy and Ling Liu. Effective mix-zone anonymization techniques for mobile travelers. *GeoInformatica*, 18(1):135–164, 2014.

[71] Xiao Pan, Xiaofeng Meng, and Jianliang Xu. Distortion-based anonymity for continuous queries in location-based mobile services. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 256–265. ACM, 2009.

[72] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[73] Judea Pearl. Heuristics: intelligent search strategies for computer problem solving. 1984.

[74] Antti Rainio et al. Location-based services and personal navigation in mobile infor-ation society. new technology for a new century. In *International Conference FIG working Week 2001, 6 11 May, Seoul, Korea*. Citeseer, 2001.

[75] Martin Raubal and Stephan Winter. Enriching wayfinding instructions with local landmarks. In *International Conference on Geographic Information Science*, pages 243–259. Springer, 2002.

[76] Kai-Florian Richter and Matt Duckham. Simplest instructions: Finding easy-to-describe routes for navigation. In *International Conference on Geographic Information Science*, pages 274–289. Springer, 2008.

[77] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 1995.

[78] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowl. and Data Eng.*, 13(6):1010–1027, November 2001. ISSN 1041-4347. doi: 10.1109/69.971193. URL `http://dx.doi.org/10.1109/69.971193`.

[79] Karin Schweizer, Steffi Katz, and Gabriele Janzen. Orientierung im raum-kognitive grundlagen und sprachliche realisierung. *Tourismus Journal*, 4(1):79–101, 2000.

[80] Jacob Shapiro, Jerry Waxman, and Danny Nir. Level graphs and approximate shortest path algorithms. *Networks*, 22(7):691–717, 1992.

[81] Jessica Smith, Allison Kealy, and Ian Williamson. Spatial data infrastructure-an integrated architecture for location based services. *Geomatics Research Australasia*, pages 67–80, 2002.

[82] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):571–588, 2002.

[83] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[84] Barbara Tversky and Paul U Lee. Pictorial and verbal tools for conveying routes. In *Spatial information theory. Cognitive and computational foundations of geographic information science*, pages 51–64. Springer, 1999.

[85] Ting Wang and Ling Liu. Privacy-aware mobile services over road networks. *Proceedings of the VLDB Endowment*, 2(1):1042–1053, 2009.

[86] Stephen Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, January 1962. ISSN 0004-5411. doi: 10.1145/321105.321107. URL `http://doi.acm.org/10.1145/321105.321107`.

[87] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. Constructing popular routes from uncertain trajectories. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 195–203. ACM, 2012.

[88] Matthias Westphal and Jochen Renz. Evaluating and minimizing ambiguities in qualitative route instructions. In *Proceedings of the 19th ACM SIGSPATIAL Inter-*

national Conference on Advances in Geographic Information Systems, pages 171–180. ACM, 2011.

[89] Xiaokui Xiao, Ke Yi, and Yufei Tao. The hardness and approximation algorithms for l-diversity. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 135–146, 2010.

[90] Toby Xu and Ying Cai. Exploring historical location data for anonymity preservation in location-based services. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

[91] Mingqiang Xue, Panos Kalnis, and Hung Keng Pung. Location diversity: Enhanced privacy protection in location based services. In *Location and Context Awareness*, pages 70–87. Springer, 2009.

[92] Man Lung Yiu, Christian S Jensen, Jesper Møller, and Hua Lu. Design and analysis of a ranking approach to private location-based services. *ACM Transactions on Database Systems (TODS)*, 36(2):10, 2011.

[93] Tun-Hao You, Wen-Chih Peng, and Wang-Chien Lee. Protecting moving trajectories with dummies. In *2007 International Conference on Mobile Data Management*, pages 278–282. IEEE, 2007.