University Library

A gateway to Melbourne's research publications

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:
Ghane Ezabadi, Soheila

Title:
Privacy-Preserving Approaches to Analyzing Sensitive Trajectory Data

Date:
2019

Persistent Link:
https://hdl.handle.net/11343/240860

Terms and Conditions:
Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.

Privacy-Preserving Approaches to Analyzing Sensitive Trajectory Data

Soheila Ghane Ezabadi

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

School of Computing and Information Systems
THE UNIVERSITY OF MELBOURNE

October 2019

Copyright © 2019 Soheila Ghane Ezabadi

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.

Abstract

The evolution of smart devices and sensor-enabled vehicles has brought forward the capability of collecting large and rich datasets. The datasets provide unprecedented opportunities for devising the next generation of location-based decision systems. Analysing detailed continually updated information of a user's status such as location, speed and direction is vital in improving the safety, reliability, mobility and efficiency of any form of location-based services in smart cities. More generally, trajectory data is paramount for studying people's movement patterns, shopping behaviour and preferences (i.e., visited cafes, parks, and their sequence of points of interest). However, such fine-grained data raises significant concerns about the privacy of individuals, which in turn hinders the further development of next generation applications that benefit from trajectory data. Such data can reveal various sensitive information about individuals such as their home and workplace locations, whereabouts over time and health. Recent approaches to address such concerns use a strong privacy guarantee – known as differential privacy. Their aim is to tackle a core privacy challenge: publishing modified datasets of individuals without compromising their privacy while not sacrificing the utility of the published data. However, the current approaches guaranteeing differential privacy are limited in scalability and utility for real applications which both are crucial for later usage or data analytics. In this thesis, we are concerned with publishing trajectory data which poses privacy risks due to its sequential nature. A key issue is that the known algorithms fail to preserve the utility of published trajectory data when perturbing it to satisfy differential privacy. Critical information of trajectory datasets such as total travel distances and frequent location patterns in trajectories cannot be fully preserved by the existing differentially private algorithms. This thesis investigates three research issues. First, it is known that simple histograms, which is widely studied under differential privacy, are insufficient to capture aggregated information for spatial data. Our first work shows how to use instead spatial histograms to provide accurate distribution of

traffic counts with differential privacy guarantee. Spatial histograms must satisfy sequential constraints (spatial) and naively applying differential privacy can destroy sequential constraints. Our proposed algorithm computes new information about trajectory counts without destroying spatial constraints and hence, improves the utility of published data.

We further refine the algorithm to improve the utility of the published data by incorporating the traffic distribution. Intuitively, dense regions gain more information about the trajectory counts compared to sparse regions. Since the density of different regions might be uneven, we need to directly use trajectory densities to accurately compute information about the trajectory distribution in the regions for efficiently scaling the added noise to ensure differential privacy. Spatial histogram data has limitations in terms of spatial queries. For example, we cannot ask queries such as "how many trajectories start from location A and end at location B?". To address this limitation, in our third work, instead of using count information from trajectories as in spatial histograms we use actual trajectory data. We introduce a graphical model to capture accurate statistics about the movement behaviours in trajectories. Using this model, our algorithm privately generates synthetic trajectories such that the noise is optimally added to capture the movement direction of a trajectory. Our algorithm preserves both the spatial and temporal information of trajectories in the generated dataset, requires less memory and computation than competing approaches, and preserves the properties of original trajectory data in terms of traveled distance, movement patterns and locations of interest. Our extensive theoretical and experimental analysis shows the significant improvement in the utility of published data generated by our algorithms.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Soheila Ghane Ezabadi, October 2019

Acknowledgements

I would like to express my deepest gratitude and appreciation to all kind individuals who supported me throughout my PhD. First and foremost, I would like to thank my supervisors, Prof. Lars Kulik and Pro. Ramamohanarao (Rao) Koragiri for their constant support, guidance and constructive comments during my PhD. I consider myself fortunate to have such patient and insightful mentors whose valuable advice have benefited me immensely during my PhD. I wish to thank their encouragement and faith in me that helped me grow as a researcher. I would also like to thank A/Prof. Sherah Kurnia for her kind support and advice as the chair and member of my advisory committee. I must also acknowledge Dr. Alireza Jolfaei for collaborating in my research and I am extremely grateful for his support, constructive suggestions, advice and encouragement.

My sincere thanks to the school of Computing and Information Systems for supporting graduate students and providing excellent conditions for research. I wish to thank the University of Melbourne for empowering student initiatives in the advancement and encouragement of women in technical domains. My experience as the chair of IEEE Women In Engineering group as well as the student ambassador of Melbourne School of Engineering has been extremely rewarding.

Special thanks go to Alex Vedernikov, Sadegh Motallebi, Shima Rashidi, Lida Rashidi, Sameera Kannangara, Lakmal Muthugama, Gayashan Amarasinghe, Yixin Xu, Li Li and Elham Naghizade for their friendship, help, encouragement and the joyful memories they made during my PhD.

Last but not least, my deepest thanks to my parents for their unconditional support, love and affection in my life. My mother has always believed in me which inspired me to work hard towards my dreams. I am indebted to my parents who thought me how to be determined in life and resilient in facing challenges. Special thanks to my brothers Saeed Ghane and Shahab Ghane for cheering me up with their words of love and support at some difficult times during my PhD.

Preface

The main contributions of this thesis are discussed in Chapters 3, 4 and 5, which are respectively based on the following published or under review papers. The thesis research has been carried out at The School of Computing and Information Systems, The University of Melbourne. I am the primary author of these papers and have contributed more than 50%.

- **Soheila Ghane**, Lars Kulik, Ramamohanarao Kotagiri (2018), "Publishing Spatial Histograms Under Differential Privacy", the 30th International Conference on Scientific and Statistical Database Management, ACM, p. 14:1-14:12. (Chapter 3)
- **Soheila Ghane**, Lars Kulik, Ramamohanarao Kotagiri, "A Differentially Private Algorithm for Range Queries on Trajectories", under submission to Journal of Knowledge and Information Systems (KAIS), Springer (invited paper). (Chapter 4)
- **Soheila Ghane**, Lars Kulik, Ramamohanarao Kotagiri (2019), "TGM: A Generative Mechanism for Publishing Trajectories with Differential Privacy", IEEE Internet of Things Journal, DOI 10.1109/JIOT.2019.2943719. (Chapter 5)

To my mother, Forouzandeh.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

PROTECTING individual privacy has become of major concern with the continuous rise of ubiquitous communication technologies. Individual privacy is a right that individuals can choose whether to share information about themselves. However, the growth of such pervasive technologies allows tracking millions of users simultaneously without their consent. The collection and persistent storing of vast amounts of personal mobility data into interconnected databases that create rich and seamless information about the individuals' daily activities which can be a serious threat to their privacy. Protecting individual privacy is ensuring that a third party is not able to (re)identify a user from a given set of collected trajectories.

Ensuring individual privacy while collecting and storing of fine-grained spatio-temporal data has become an urgent research issue in recent years. A number of generic approaches have been proposed which promise privacy through aggregation or randomization the data. Among those approaches, *differential privacy* has emerged as a particularly prominent model because it offers a proven guarantee on an individual's privacy by randomizing the data with the aim of promising utility.

Whilst spatial and spatio-temporal data can be seen as generic data, and thus it seems tempting to simply apply differential privacy to such datasets, this often leads to a significant loss of data utility. The reason is that spatio-temporal data has its own challenges. For example, given a GPS trajectory of a user traveling along a coastal road, a simple randomization algorithm might perturb the GPS points in such a way that most of the user's locations are in the sea, leading to little utility. Adding time to a sequence of locations (i.e., trajectory) makes the randomization more challenging as the starting location cannot happen after the destination location. More generally, trajectory data, is highly distinctive, which makes it hard to preserve its utility while hiding identifiable

information under differential privacy.

Trajectory databases have indispensable tools of our daily lives as they provide better transportation analysis, enable personalization through relevant location-based advertisements and help in resolving liability attribution for a traffic accident. Due to the nature of trajectories, a naive or ineffective privacy protection model can easily reveal sensitive personal information about users. A fitness tracking app, for example, inadvertently disclosed the location of secret US army based on soldiers' trajectories [1]. Furthermore, demand-side platforms for targeted advertising in mobile phone apps paving the way to uncontrolled collection of personal trajectories [2], and iOS devices used to store raw trajectories and send them to Apple [3].

Regardless of the intended application of data processing, preserving individual privacy is a major concern in trajectory data manipulation. In this thesis, we address this vital and yet open problem in releasing complete spatio-temporal datasets to third parties: promising *utility* under differential privacy, as a strong guarantee on individual privacy. Ensuring utility along with privacy is of critical importance, especially in the latest development of mobile phones where the fine-grained location of users are continuously collected by service providers [4] and shared with third parties.

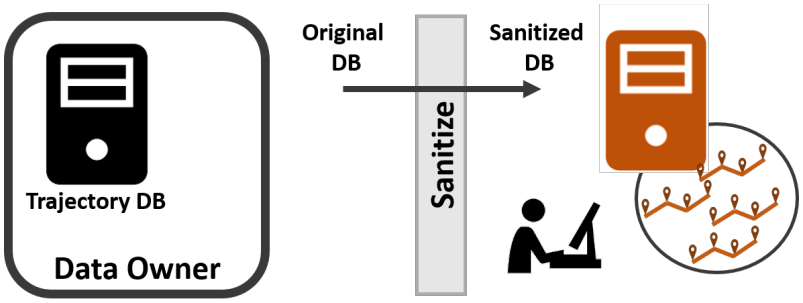
1.1 Motivation

The ever-increasing connectivity using wireless technology, data sharing, social media, the willingness to post data, and the use of mobile technology as a commodity suggest the urgent need of protecting individual privacy. In fact, services based on personal data records promise to be life-changers for the newer generations, with a clear trend of innovation happening in the data domain, where the exponential quantity of user's data is being collected and exploited by third parties. *Location* and *time* of daily actions form key components of this data, pervasively used to make personalized services. Most of these services are often rely on *trajectory data*, i.e., information about *spatio-temporal trajectories* of single individuals. A trajectory is a two-component entity – location and time – with a sequential order between events. In applications that only focus on the *locations* and *order* of events, a trajectory can be simplified as a sequence of geographical locations where the time order is implicitly represented by the sequential order in locations.

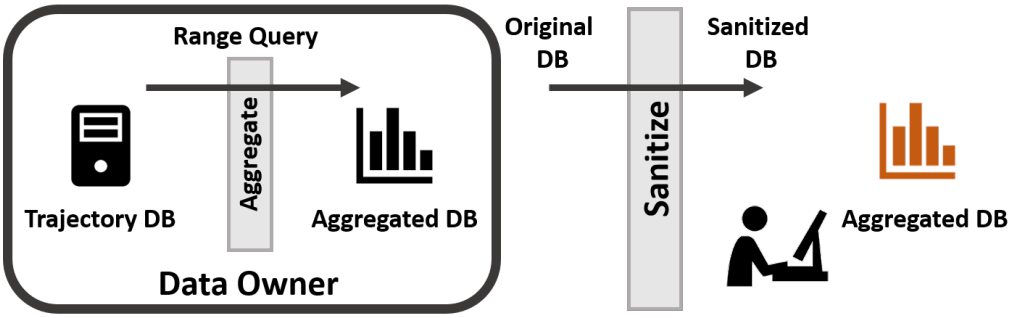
The source of trajectory data can be for example location-based service (e.g., Google Maps, FourSquare, games like Pokemon Go, and recently added feature in generic platforms such as Twitter and Instagram) for service operation; cellular network operators for purposes including billing, traffic engineering or added-value service development; or modern car navigation systems for determining the congestion level of roads by navigation system providers or determining liability in case of accidents by insurance companies or profiling driving styles and associated risk levels. Such new and pervasive technologies allow the collection of trajectory data in very large scales, which in hence, enable many applications to track and know the movement patterns of millions of individuals. These applications encompass a variety of contexts, including intelligent transportation, assisted-life services, urban planning, location-based marketing, data-driven decision-making, or infrastructure optimization. As a result, network and service providers are gathering, storing, provisioning databases of trajectories, and circulating and trading them with third parties.

Due to the nature of trajectories, especially considering the ever-increasing interest in storing and sharing trajectory, datasets calls for *privacy preserving trajectory publishing*. This calls for an approach that transforms trajectory data before releasing it to a third party such that the data remains useful (i.e., utility) while individual privacy is preserved. A common practice in privacy-preserving trajectory publishing is to publish aggregated mobility data, such as the number of users covered by a cellular tower at a specific timestamp. Unfortunately, aggregation cannot guarantee a strong level of protection. Recent studies on datasets of large scales demonstrated the significant risk of privacy breach given the aggregated mobility data without any prior background knowledge [5] [6]. Developing an algorithm that provably prevents any personal information inference from the original trajectory dataset and at the same time ensures the utility of released data is a tremendous challenge and has drawn substantial research efforts over the past decade.

To address this challenge, recent approaches develop algorithms based on *differential privacy* [7], a principal privacy model that provides a mathematically proven guarantee on the individual privacy by randomizing data. The strong guarantee of differential privacy has made large cooperates including Google [8], Apple [9], Uber [10] to develop differentially private algorithms for storing and sharing of user data.



(a) Data owner published the original trajectory dataset to a third party.



(b) Data owner computed aggregated statistics and publishes the aggregated data to a third party.

Figure 1.1: The two main modes in privacy preserving publishing of trajectory datasets.

1.2 Challenges and Main Goals

While differential privacy ensures individual privacy, the nature of trajectory data makes it hard to preserve the utility of published data. In fact, a trajectory is a sequence of timestamped locations where the continuous time-evolving nature of trajectories imposes a great impediment on direct application of differential privacy to spatio-temporal datasets.

Take for instance a scenario in Figure 1.1-a where a data owner has stored a fine-grained trajectory of connected vehicles for navigation purposes and intend to share the dataset with a third party for traffic management analysis. It has been shown that randomizing each location in a trajectory is vulnerable to inference attacks since an adversary is able to refine the obfuscated locations based on its correlation with speed and direction of consecutive location points. Additionally, the underlying road network plays an important role in shaping the movement pattern of the connected vehicles. For example, an adversary can remove the noise from the perturbed trajectory by mapping the locations into a given road network. When applying differential privacy,

Chapter	Mode	Query Type	Challenge
3	aggregated data	range query	ensuring consistency of generated data and efficiency of mechanism
4	aggregated data	range query	optimizing the generated data considering data and query properties
5	trajectory data	any query type	developing a spatial structure to capture statistics and private mechanism to trajectory generate dataset

Table 1.1: Thesis objectives and challenges.

hiding such correlations in trajectory may impede extracting useful knowledge due to the loss of data utility. In the example of the connected vehicle, large fluctuations in speed due to the added noise for protecting privacy may mislead the routing mechanism in the data collector.

Figure 1.1-b shows another data publishing scenario wherein the data owner publishes aggregated mobility data rather than original trajectory dataset. Such *aggregated data* might be managed for particular applications. An example of aggregated data which is of paramount importance is the number of distinct trajectories in various regions throughout the spatial space. A wide range of applications rely on this aggregated data, including location-based marketing, targeted advertising, or transportation analysis. A common trait to these applications is that they are all based on a specific query type named *range query* which counts the number of distinct trajectories in a specified region. Publishing aggregated count of trajectories throughout the spatial space enables the applications to use the data for efficiently computing arbitrary range queries. To publish such aggregated counts under differential privacy, the mechanism can exploit the data or query properties in reducing the scale of distortion to satisfy privacy and hence, achieve higher utility.

The main goal of this thesis is to propose mechanisms that protect individual privacy in publishing trajectories or simply trajectory privacy. The privacy challenge is studied when the original trajectory dataset is being published to a third party (Figure 1.1-a), and also when the data owner computed aggregated mobility data and shares it with third parties for particular applications (Figure 1.1-b). We explore preserving the utility of data while satisfying privacy with computational efficiency as a crucial feature in our introduced approaches. The focus of this thesis is on differential privacy, a strong privacy model ensuring the released data is independent of an individuals datum, i.e., given almost all other data, the adversary cannot infer an unknown individual's datum.

Three main challenges in trajectory privacy are addressed in this thesis: how to privatise answering range queries on a given trajectory dataset, how to optimize the aggregated data for range queries given a query set and a trajectory dataset, and how to publish a trajectory dataset privately, Each question addresses one of the challenges in trajectory privacy when publishing original trajectories or aggregated data. These questions are further explored and discussed in the next section as the main research questions. Table 1.1 summarises the thesis objectives and corresponding technical challenges.

1.3 Key Research Questions

This section outlines the main research questions of this thesis. A summary of each research question is provided with the core contributions of our work.

1.3.1 Differentially Private Publishing of a Spatial Structure for Answering Range Queries

The main question explored in Chapter 3 is privately and efficiently answering of range queries on trajectories. Range query is a key query in understanding trajectory data and offering location-based services. This query computes the total count of trajectories contributed to the query area, for example, the total number of cars passing through a road segment covered by the query area, or the number of smartphone users in each block of a city. The range query is also known as aggregate query or distinct count query, and has been extensively studied in the (spatial) data management literature [11], [12], [13].

To correctly answer range queries, spatial histograms are popular data structures [14], [15], [16] that efficiently store trajectory data. A spatial histogram decomposes the underlying space into finite cells. Each cell captures the number of trajectories intersecting with the cell. In contrast to simple histograms where cell counts are independent, in spatial histograms, there is a strong dependency between the cell counts. A trajectory may overlap a sequence of cells and thus, adding/removing the trajectory leads to changing the values of all contributing cells. In a spatial histogram, changing one cell value may change a sequence of adjacent cells. This sequentiality constraint is the key property of spatial histograms that

enables them to answer range queries on trajectories. We will use Distributed Euler Histograms (DEHs) [14] as spatial histograms as they can count the unique number of trajectories by keeping trajectory counts for each cell as well as counts for each trajectory intersections of edges between two adjacent cells.

Despite the importance of spatial histograms in research and commercial applications, data providers need to take privacy concerns into consideration before publicly releasing data. Even though a histogram aggregates trajectory data, an individual's identity might still be inferred. Xu et al. [5] showed that up to 91% of trajectories could be uniquely recovered in aggregated movement trajectories. This problem is even more severe in rural areas with a low density of trajectories. However, anonymizing a spatial histogram is difficult, as changing a cell value, randomly and independent of its dependency with adjacent cells, may violate the consistency of values and hence, compromise the utility of anonymized histogram.

A robust and strong privacy model, such as differential privacy [7] is needed to guarantee the privacy of a published histogram. In addition, the anonymization mechanism needs to ensure the consistency of published histogram regarding the cell dependency and high accuracy in answering range queries.

Contributions

In Chapter 3, we propose a Private Spatial Histogram (PriSH) mechanism that outputs a synthetic spatial histogram with differential privacy guarantees. To the best of our knowledge, we are the first to address the problem of privately publishing spatial histograms. In the synthesis process, PriSH utilizes the data dependency and ensures the consistency of cell values in the published histogram. This process exploits range queries in optimizing the accuracy of the generated histogram and ensures high accuracy in answering spatial range queries. We summarize our contributions in Chapter 3:

- We propose PriSH for publishing spatial histograms with differential privacy guarantees.
- We develop a private mechanism that utilizes data properties in synthesizing the data and ensures the consistency of cell values in the published histogram (in Chapter 4 we optimize the published histogram for the given set of range queries).
- Our mechanism ensures high accuracy in answering spatial range queries.

1.3.2 A Data- and Query-Aware Algorithm for Range Queries Under Differential Privacy

Chapter 4 presents a Data- and Query-Aware Mechanism (DQAM) that synthesizes spatial histograms by tuning the added noise to the data and query properties. A naive approach adds an equal noise to the cell counts in the histogram to ensure privacy. Since a trajectory may intersect multiple cells, the cells are sequentially dependent. Thus, naively scaling noise to satisfy differential privacy can violate the sequentiality between the cells, which is crucial in representing trajectories. Moreover, the sensitivity of cell counts to the added noise varies depending on the density of trajectories in different regions of data space. A large region of space might be sparse with small cell counts which makes the cell counts in the region highly sensitive to the added noise. In contrast, a region might be small but very dense such that cell counts are large and highly noise resistance. The proposed approach should be able to ensure maintaining the data properties (called data-awareness) in the synthesized data. Additionally, our approach makes use of the given set of range queries to learn the distribution of data (called query-awareness).

Similar to Chapter 3, DQAM takes a spatial histogram, and a set of range queries as input and uses the correlation between the queries (i.e., is query-aware) to synthesize the histogram. In contrast to Chapter 3, DQAM identifies the density of different regions in data and utilizes the density in scaling the added noise to the generated synthetic histogram (i.e., is query-aware). Also, DQAM generates the optimal synthetic histogram in each iteration of the synthesizing process that satisfies the sequentiality of cells. The proposed mechanism in Chapter 3 applies a greedy strategy to satisfy the sequentiality requirement among histogram cells, which may lead to overcorrection. While the work in Chapter 3 successfully deals with trajectories uniformly distributed in the data space, it can result in considerable loss of utility for trajectory data sets with a non-uniform distribution in different regions. DQAM is designed to deal with such realistic trajectory distributions by capturing the density of trajectories in different regions and using the density as a weighting measure: each region is assigned a weight relative to its density. To guarantee differential privacy, DQAM uses the well established Laplace mechanism [7] and Exponential mechanism [17] for adding noise in all parts of its computations.

Contributions

We summarize the contributions of Chapter 4 as follows:

- We propose a differentially private Data- and Query-Aware Mechanism (DQAM), a mechanism that publishes a spatial histogram to answer range queries efficiently. To the best of our knowledge, DQAM is the first data- and query-aware mechanism for range queries on sequential spatial data that reduces the error by a factor up to 7.4 compared to the current approaches on trajectory data sets.
- We design an efficient data-aware algorithm with $nlog(n)$ time complexity in the number of cells in the spatial histogram that partitions the histogram into uniform regions.
- We present a query-aware and differentially private strategy that captures trajectory sequentiality and synthesizes accurate output using the given queries and partitions.

1.3.3 Publishing Trajectories with Differential Privacy

Considering the significant importance of large datasets of trajectories in many applications, Chapter 5 proposed a mechanism to publish such datasets with a promising high utility for further analytical tasks. Recent studies have developed ε-differentially private mechanisms for publishing trajectory datasets [18], [19], [20], [21] with strong privacy guarantees [22]. ε refers to the privacy level, where a lower value provides a stronger privacy guarantee. The key challenge in developing a differentially private mechanism is to preserve an accurate trajectory pattern when adding noise to ensure privacy. A naive mechanism that scales the noise proportional to the length (i.e., number of locations) of a trajectory can adversely diminish its utility. Existing mechanisms first pre-process trajectories by mapping them to a grid, then extract the most frequent patterns from these coarsened trajectories (grid cells), afterwards build a probabilistic model using the patterns with added noise, and finally post-process the noisy model to make the trajectories consistent. This model is then used to generate synthetic trajectories.

However, the state-of-the-art mechanisms fail to preserve the trajectory patterns in their generated data accurately. The pre- and post-processing steps in those mechanisms can distort the trajectory properties such as traveled distance and frequent trajectory patterns. In addition, the efficiency of existing mechanisms are highly sensitive to the size of corresponding spatial space.

The running time and the memory required increases exponentially with the number of grid cells.

Furthermore, the state-of-the-art mechanisms cannot preserve semantic properties of trajectories such as stay locations where people may spend substantial periods of time. However, such semantic properties are essential for many applications. For example, the amount of time an individual has stayed in a location identifies the place of interest for that person. Such places of interest reveal significant information for understanding the daily habits of individuals [23], [24]. Preserving the semantic properties of trajectories under differential privacy is a gap in the current literature.

Contributions

We propose a novel mechanism in Chapter 5 for publishing large trajectory datasets under ε-differential privacy. Our key contributions are:

- Effectively encoding trajectories as a graphical generative model. In contrast to the state-of-the-art model, our proposed model is data-adaptive, which leads to high scalability and can accurately capture the properties of trajectories such as traveled distance or movement patterns.
- Introduction of a highly efficient algorithm for generating trajectories under ε-differential privacy that is more than 4000 times faster than the state-of-the-art algorithms while reducing memory requirements by a factor 50.
- In contrast to existing work, our approach can generate trajectories of arbitrary length while maintaining travel distance and frequent trajectory patterns. To the best of our knowledge, TGM is the first mechanism that also captures the stay locations of trajectories.
- Our extensive experiments show that our mechanism improves the utility of published trajectories typically around an order of magnitude without making further assumptions such as the trajectory length or the velocity of objects.

1.4 Organization of the Thesis

The remainder of this thesis is structured as follows:

Chapter 2, provides a background on differential privacy and related works to privacy-preserving collecting and sharing of trajectory datasets.

Chapter 3, examines the problem of answering range queries on trajectories and introduces PriSH mechanisms to address this problem. The mechanism publishes a differentially private spatial histogram. This chapter is based on the following article:

- Soheila Ghane, Lars Kulik, Kotagiri Ramamohanarao, "Publishing spatial histograms under differential privacy", Proceedings of the 30th International Conference on Scientific and Statistical Database Management, SSDBM'2018, pp 14:1-14:12.

Chapter 4, presents a data- and query-aware mechanism for optimizing range queries under differential privacy. This chapter is based on the following article:

- Soheila Ghane, Lars Kulik, Kotagiri Ramamohanarao, "A Differentially Private Algorithm for Range Queries on Trajectories", Journal of Knowledge and Information Systems, Springer, 2019, Vol. 61, No. 3.

Chapter 5, proposes a generative approach for publishing trajectory datasets under differential privacy. We introduce a data structure to maintain movement statistics and develop an algorithm for synthesizing the trajectory dataset using the data structure. This chapter is based on the following article:

- Soheila Ghane, Lars Kulik, Kotagiri Ramamohanarao, "TGM: A Generative Mechanism for Publishing Trajectories with Differential Privacy", IEEE Internet of Things Journal, 2019, Vol. 7, No. 1

Chapter 6, summarizes the remarks and key findings of this thesis and provides potential future research direction of this research.

Chapter 2
Background

PREVENTING the disclosure of individuals' sensitive information based on published datasets has led to a concept called privacy-preserving data publishing. The main goal in privacy-preserving publishing data is anonymizing datasets such that given an anonymized dataset, the adversary cannot infer any new information about a particular individual. In this chapter, we review anonymization models and specifically, differential privacy as a leading framework with a provable guarantee on individual privacy. Anonymization models attempt to formalize individual privacy based on data context and the adversaries' background knowledge. These models enable the data owners to release sensitive data of individuals without violating individual individual [25]. This chapter reviews the issue of individual privacy in publishing datasets, followed by introducing early privacy models and their limitations. Differential privacy is then introduced with detailed discussion on its properties in ensuring individual privacy. Recent algorithms proposed for publishing datasets of locations and trajectories are later described, and their features are contrasted.

2.1 Introduction

In privacy-preserving data publishing, a database is often seen as a table of records and each record represents a person's information, and columns are specific features of that person. The purpose of data publishing is releasing information such as aggregated statistics about personal data without threatening the privacy of involved individuals. Hence, such aggregated database is also called statistical database.

Privacy-preserving data publishing models can be categorized into two types, syntactic models and perturbation models. Syntactic models apply techniques such as generalization (replacing the private value with a less sensitive feature) and suppression (not releasing the sensitive record/feature) on records of data to generalize database entries until they satisfy some conditions.

The rationale behind syntactic models is transferring the syntax of the original data to meet the requirements of a particular definition of privacy.

Suppose Alice is a patient who has *HIV* and regularly consults with specialists in a medical center. If we define privacy breach as identifying Alice as a patient in this medical center, the anonymized database should prevent the adversary (malicious user) from recognizing whether her data is the published database. The fact that Alice visits this medical service may not be considered sensitive, but the adversary has not found out that Alice has *HIV*. Then the syntactic model should generalize the syntax of data so that guarantees the released database protects this sensitive information from the adversary. Syntactic privacy models and the seminal works in this category are described in the following section.

Despite the enhanced privacy that syntactic models provide, later studies pointed out the limitations of these models and their vulnerability to different attacks [25]. To address these weaknesses, perturbation models add noise to data to hide the presence or absence of an individual or bound the probability of identifying sensitive information about an individual. Perturbation models primarily focus on tabular statistics where each cell denotes the frequency of individuals who have common values in the specified features. Privacy models in this category offer a guarantee on privacy but with a higher cost in utility.

In the above example, consider a doctor who needs to know the number of patients with *HIV* in the medical center to order the right amount of drugs. Suppose the true answers is 5, but due to perturbation, the doctor gets 8; which cause choosing a different strategy for the treatment. Such changes in the answer, especially in applications like healthcare, raises the problem of low utility and ultimately leads to wrong decisions. In addition, if the noise is added just in a random way (with zero mean), the adversary would be able to approximate the true value by asking many queries and then averaging the noisy answers.

As a consequence, the challenge of perturbation mechanisms is formalizing the noise ratio to guarantee privacy while providing useful data for further analysis. Differential privacy is a perturbation mechanism that proposes a formal mathematical guarantee on privacy which is resilient to linkage attacks and quantifies the level of the individual privacy. Many publishing mechanisms in the research community have applied differential privacy to release data.

Notwithstanding distinguished improvements in the accuracy of published data based on dif-

ferential privacy, most of these mechanisms suffer from the low utility of results and do not preserve properties of original data such as the correlation between attribute. In the second section of this chapter, we describe differential privacy and its theoretical notions in details. The last section explains the mechanisms for differentially private data publishing and explains their strong points and weaknesses.

2.2 Syntactic Privacy Models

Herein, we describe the most important syntactic privacy models for publishing data. This will clarify why models in this category are susceptible to various attacks and then, how differential privacy can overcome this vulnerability.

Generally, a database can be considered as a set of records with the following disjoint features:

- *Explicit Identifier*, which identifies the owner of a record.
- *Quasi Identifiers*, which are a set of features. Each feature in this set cannot identify the owner of a record explicitly. But when considered as a group of features, they could potentially single out the owner of the record.
- *Sensitive Attribute*, that the owner of a record does not like an unauthorized person access to this information.
- *Non-sensitive Attribute*, that could be revealed and the owner of a record is not concerned about them [26].

For publishing a dataset while preserving the privacy, it is clear that *explicit identifiers* should be removed. However, just removing this kind of feature is not enough and cleverer operations are needed to guarantee the adversary would not identify an individual or sensitive information [27].

The Massachusetts government agency and re-identifying William Weld, former governor of this state, from the released data is a real-life proof of privacy breach. They released medical records for all state government employees and made sure that the data is anonymized so that confidentiality of individuals involved is protected.

Actually, they removed all *explicit identifier* features, such as social security number, name and address. However, it was not enough to protect individuals' privacy. To show this weakness, Sweeney [27] matched this data against public voters list for Massachusetts and could single out

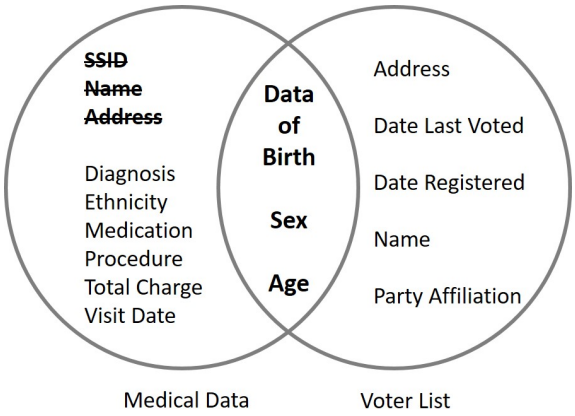


Figure 2.1: Linkage attack after removing *explicit identifier* features.

William Weld's medical record (Figure 2.1).

2.2.1 *k*-Anonymity

Even though by removing *explicit identifier* features the adversary cannot search an individual to find her/him in the database, the set of *quasi identifiers* makes it possible to re-identify an individual in the data. In the above example, the adversary knew that William Weld lives in Massachusetts. Then by getting access to auxiliary information, public voters list, and matching "date of birth", "age" and "sex" features in two datasets, could exactly find his medical record (Figure 2.1). In this case, "date of birth", "age" and "sex" are *quasi identifiers* that together could identify the owner of a record. Above simple privacy model could not protect the privacy of involved individuals against such *linkage attacks* and we need more sophisticated privacy models that are provably correct in ensuring privacy.

To prevent this type of linkage attack, *k*-anonymity [28] was proposed:

A table of data is k-anonymous if for each set of quasi identifiers, there is at least k-1 other records with the same values for their quasi identifiers.

The intuition behind this privacy model is hiding a record in a group of record with the same *quasi identifiers*. Hence, if an adversary could link the external knowledge to a k-anonymous dataset to find an individual, in the best case, a group of records with size k will be recognized.

For example, suppose a medical center has a dataset including patients medical information (Figure 2.2 (a)). After removing *explicit identifier* features like "Name", "Address" and "SSID",

(a) Patients Dataset

Job	Sex	Age	Disease
Mechanical Engineer	Female	45	HIV
Computer Engineer	Female	49	HIV
Business Lawyer	Female	38	HIV
Labor Lawyer	Male	33	Heart Attack
Dancer	Male	35	HIV
Painter	Male	30	Flue

(b) Public Dataset

Name	Job	Sex	Age
John	Dancer	Male	22
Alice	Computer Engineer	Female	49
Bob	Painter	Male	31
Fred	Labor Lawyer	Male	36
Cathy	Computer Engineer	Female	54
Emily	Mechanical Engineer	Female	47

(c) 2-Anonymous Dataset

Job	Sex	Age	Disease
Engineer	Female	[40 – 50)	HIV
Engineer	Female	[40 – 50)	HIV
Lawyer	Female	[30 – 40)	HIV
Lawyer	Male	[30 – 40)	Heart Attack
Artist	Male	[30 – 40)	HIV
Artist	Male	[30 – 40)	Flue

(d) Dataset with 2-Diversity

Job	Sex	Age	Disease
Professional	F/M	[30 – 50)	HIV
Professional	F/M	[30 – 50)	HIV
Professional	F/M	[30 – 50)	HIV
Professional	F/M	[30 – 50)	Heart Attack
Artist	Male	[30 – 40)	HIV
Artist	Male	[30 – 40)	Flue

Figure 2.2: Examples to illustrate susceptibility of syntactic models to linkage attacks.

they want to release the data based on k-anonymity privacy model. Considering the set of "Job", "Sex" and "Age" as *quasi identifiers*, according to k-anonymity the value of *quasi identifiers* generalize to a level to have at least k records with the same values for "job", "sex" and "age".

Figure 2.2 (c) shows a 2-anonymous version of original data where values of attribute "job" are generalized to *Engineer*, *Lawyer* and *Artist*, and values of attribute "Age" are generalized into distinct intervals. Before applying k-anonymity an adversary could join the data to a public dataset like Figure 2.2 (b) and re-identify *Alice* who is *Computer Engineer*, *Female* and *49 years old*. From the 2-anonymous table the attacker finds two records with the same *quasi identifiers* and would not be able to single out her record. In this privacy model, the parameter k is identified by the data curator and depends on the expected level of anonymity.

2.2.2 *l*-Diversity

Hiding an individual's record in a crowd of similar record does not always satisfy privacy requirements of the dataset. In the above example, the attacker still can infer with 100% that Alice has "HIV" for her *sensitive attribute* "Disease". Because after generalizing, all the records in her group

of *quasi identifiers* have the same value for their *sensitive attribute*. Hence, k-anonymity and its variants such as Multirelational k-anonymity [29] and (x,y)-anonymity [30] cannot prevent inferences to the *sensitive attributes* of an individual.

l-Diversity [31] is a privacy model that formalizes this kind of linkage attacks to *sensitive attributes* of involved individuals. With respect to this notion of privacy:

A table satisfies l-diversity privacy requirements if in each quasi identifiers group there is at least l distinct value for sensitive attributes.

In our example, Figure 2.2 (d) shows a dataset with 2-diversity as each group of *quasi identifiers* has two distinct values for attribute "disease". Due to properties of l-diversity, this dataset automatically satisfies 2-anonymity [32].

l-Diversity and its variants like (c,l)-diversity [31], t-closeness [33] [34] or (α,k)-anonymity [35] focus on frequency of distinct values in *sensitive attributes* and formalize an adversary's background knowledge to guarantee privacy. However, these privacy models limit the *auxiliary information* that the adversary could get access to infer information about individuals.

Name	Job	Sex	Age
John	Artist	Male	[30 - 40)
Alice	Engineer	Female	[40 - 50)
Bob	Artist	Male	[30 - 40)
Fred	Lawyer	Male	[30 - 40)
Cathy	Engineer	Female	[40 - 50)
Emily	Lawyer	Female	[30 - 40)
Tony	Artist	Male	[30 - 40)
David	Lawyer	Male	[30 - 40)
Gladys	Engineer	Female	[40 - 50)

Figure 2.3: An example of public data which is 3-anonymous.

2.2.3 δ-Presence

Both k-anonymity and l-diversity models, assume the adversary already knows that the victim's record is in the dataset. Then, try to formalize privacy so that the adversary cannot infer new information by linking *auxiliary information* against the released data. Nevertheless, the adversary usually is not sure about the presence or absence of an individual's information, and a data curator wants to assure that releasing the data does not help the adversary to find out this information.

In the example above, if the adversary gets access to another public dataset in Figure 2.3, by matching it against the anonymized dataset in Figure 2.2 (c), he/she could infer with probability of $\frac{2}{3}$ that Bob is in the released data. Because the *quasi identifiers* group with values "Artist, Male, $[30-40)$" has two records in Figure 2.2 (c) and the public dataset in Figure 2.3 has three records with these value for attributes "Job", "Sex" and "Age".

To prevent this type of linkage attack δ-*Presence* [36] privacy model has been proposed that bounds the probability of privacy breach due to a specific auxiliary information. As stated in this privacy model:

A table B' has $(\delta_{min}, \delta_{max})$-presence privacy if for all $t \in A$ it has $\delta_{min} \leq P(r \in B|B') \leq \delta_{max}$ where B is the original data, B' is the anonymized table, and A is a public dataset so that $B \subseteq A$.

This privacy model ensures that by releasing the data, an adversary would not be able to infer new information about an individual. This guarantees could also secure an individual's record (linkage attack to *quasi identifier*) and sensitive information (linkage attack to *sensitive attributes*). However, δ-presence assumes the data curator is already aware of possible auxiliary information (in our case, dataset A) that the adversary might use; which restricts the practicality of this privacy model.

2.2.4 Discussion: Critical Weaknesses of Syntactic Models

Syntactic privacy models limit the background knowledge of an adversary in linkage attacks by making some assumption about available auxiliary information. In general, such information is not available and therefore releasing information with adequate privacy is not practical. As a result, there is no guarantee that the privacy model is practical for real applications. Moreover, syntactic models are based on two categories of features: (1) *quasi identifiers* and (2) *sensitive attributes*. This categorization is non-trivial and might be different depending on the application. Syntactic models assume that a data owner is able to categorize the feature, which is not always a correct assumption. Furthermore, there is a wide variety of privacy definition used in syntactic models and also, the types of attacks that the models are preventing. Given such a variety, it is very challenging for data owners to decide which model suits most to their data and their expectations about protecting individual privacy.

Data owners need to have a formal definition of privacy. The privacy model should propose specific mechanisms to be practical regarding adversary's background knowledge and possible linkage attacks. It should not restrict the data publisher to data-specific parameters such as identifying types of attributes, number of expected records to have the same *quasi identifiers*, number of distinct values for each *sensitive attribute*. Differential privacy is a model that addresses these challenges by providing a general definition of "individual privacy" and also, a strong guarantee on ensuring the individual privacy regardless of adversaries background knowledge.

2.3 Differential Privacy

Beside syntactic privacy models, perturbation privacy models do not focus on data-specific parameters to define and preserve the privacy of individuals, but instead, add noise to limit the adversary's gain by accessing to the released data. In this category, privacy models randomly perturb the data values to hide sensitive information of individuals.

However, studies show that most of the perturbation mechanisms are predictable and cannot provide a guarantee on data privacy [37]. Still, other privacy models in this category simplify the problem to limited background knowledge of the adversary [38] and then cannot provide privacy guarantee beyond their assumptions. Another challenge in perturbation privacy models is the utility of released information. Adding random noise to reduce data disclosure may cause meaningless outputs that do not contain useful information about the original data.

The limitations of syntactic and perturbation privacy models stem from their definition of privacy that they have attempted to formalize. Assuming an individual's record is in the dataset, the adversary's knowledge before and after accessing the data is compared, and any improvement on this knowledge has been considered as information disclosure. Actually, these models follow the privacy goal provided by Dalenius [39]:

"In 1977 the statistician Tore Dalenius articulated an ad Omnia (as opposed to ad hoc) privacy goal for statistical databases: Anything that can be learned about a respondent from the statistical database should be learnable without access to the database."

Consider an example where we analyze the anonymized database and find out *smoking causes cancer*. This finding is the knowledge that the adversary learns by data and may affect an individual

whose information is not even in the database. In fact, each person that smokes might be affected by this finding and is not related to the individual's involvement in the database. However, due to the Dalenius's definition, the utility of released data are directly related to the privacy leakage. As a consequence, all the privacy models based on this definition try to model auxiliary information and information that the adversary might be able to learn from data. Nevertheless, there is a wide variety of auxiliary information that makes it impractical to define prior and posterior knowledge of an adversary formally.

To address the dilemma of auxiliary information and the adversary's knowledge, Dwork [40] proposed a new definition of privacy that surpasses auxiliary information, *differential privacy*. In this view, privacy is defined by the probability distribution of data with or without an individual's record. If removing the record causes a large variation in the probability of output, it would be a privacy breach, and differential privacy aims to answer this threat. In other words, differential privacy does not focus on anonymizing the database but instead focuses on controlling accesses to the data. Hence, differential privacy assures that data analyzing mechanisms can get useful aggregate information from personal data and also, provides provable guarantee about the privacy of involved individuals.

2.3.1 Definition and Building Blocks

We begin with formalizing the concept of dataset and neighborhood in datasets. Then we formalize differential privacy and provide its interpretation.

Consider a relational schema $R(\mathbb{F})$ with $\mathbb{F} = \{F_1, F_2, ..., F_k\}$. The data universe $\mathscr{T}$ defines as a cross product of domain of features in $dom(\mathbb{F}) : \{dom(F_1) \times dom(F_2) \times ... \times dom(F_k)\}$. Where a $dom(F_i)$ can be discrete or continuous, finite or infinite, or ordered or unordered. Each element in $\mathscr{T}$ can be considered as a *record* r that expresses a distinct combination of values in $dom(\mathbb{F})$. Typically, a dataset D is an instance of $\mathscr{T}$ that its features are a projection of F. Each record r in D represents the count of elements in the instance with type r.

We formally define a statistical database as a function $D : \mathscr{T} \to \mathbb{N}$ where $\mathbb{N} = \{0, 1, 2, ...\}$ denotes the count of elements for each type of $r \in \mathscr{T}$. Therefore, the size of a database computes as $\|D\|_1 = \sum_{r \in \mathscr{T}} |D(r)|$. Accordingly, we may have many instances of D from the universe $\mathscr{T}$. All possible instances are formalized as $\mathscr{D} : \mathscr{T} \to \mathbb{N}$ and $\mathscr{D}_{n,m} = \{D \in \mathscr{D} \mid |D| = n, |\mathbb{F}_D| = m\}$

denotes all datasets with m features/dimensions and n records.

Consider a medical database with only one attribute *Disease*. Assume the medical center is only interested in patients with *"Flu"* and *"HIV"*. In this case, the universe $\mathscr{T}$ consists of two elements $\{Flu, HIV\}$ and the database contains at most two record with the number of patients for that disease. For example, let the database D contain two records, one for *Alice* who has *HIV* and one for *Bob* with *Flu*. Now, suppose another patient *Emily* with *Flu* joins the database of patients. So, we have a new database D' that differ just in one record with the database D and we call them *adjacent* or *neighbor* databases.

Definition 2.1. *(Neighbor Databases) Two databases $D, D' \in \mathscr{D}$ are called neighbor if*

$$\|D - D'\|_1 = \sum_{r \in \mathscr{T}} |D(r) - D'(r)| \leq 1$$

That is, one database could be generated by adding or removing one individual in the other database. Another interpretation of Definition 2.2.1 is stronger and considers two databases that differ in at most one value (changing the value of one record, not adding or removing one). Each interpretation leads to a different sensitivity for queries (discussed next). In this thesis, we use the first interpretation for neighbor databases.

Usually it is more convenient to define a database as a vector. Formally, we can define a vector v over the universe $\mathscr{T}$: $v \in \mathbb{N}^{|\mathscr{T}|}$; in which each element $v(r)$ represents the number of individuals in vector v of type $r \in \mathscr{T}$. An ordered vector v represents database D if for each $r \in \mathscr{D}$ we have $v(r) = D(r)$. In this case, two database D, $D' \in \mathscr{D}$ are neighbor if and only if the corresponding vectors v and v' have distance $\|v - v'\|_1 \leq 1$. In cases that D has just one dimension, the data vector v denotes a *one-dimensional histogram*. In other cases, the data vector v indicates a *multi-dimensional histogram*.

A privacy mechanism should be able to hide such differences between two neighbor datasets that are different only in one record. To explain, suppose the adversary knows Alice and is interested in finding out "what Alice's grade is in the Math subject". Assume her grade is 85. And there are exists two neighbor datasets D and D' that the adversary cannot access but can ask a query. Let us assume that the adversary knows D' does not contain Alice's record and asks the query "How many students have grade more than 80 in Math?". As the query in this example is a *deterministic function*, the answer on dataset D let us say it might be 5 and on dataset D' 6. Then, the adversary

can conclude that Alice's grade is more than 80. This is a privacy breach since our aim is hiding personal information regardless of an adversary's prior knowledge. In particular, differential privacy needs a kind of randomness for preventing such privacy breach, and deterministic functions cannot provide it.

Deterministic functions therefore cannot comply with privacy requirements of differential privacy. Hence, *random functions* play a key role in this privacy model. In differential privacy we are interested in *random functions* or *random mechanisms* that have discrete probability distributions in the range of outputs.

Definition 2.2. *(Random Mechanism) A random mechanism is a function $\mathscr{M}$ that maps an input dataset D into a probability distribution ρ over a range $\mathscr{R}$:*

$$\mathscr{M} : D \to \rho(\mathscr{R})$$

Typically, a mechanism can be any random function that performs some operations on database D and generates output $\mathscr{S}$. However, differential privacy is interested in mechanisms that bound the difference between outputs over neighbor datasets:

Definition 2.3. *(Differential Privacy)[40] A mechanism $\mathscr{M} : D \to \rho(\mathscr{R})$ satisfies (ε, δ)-differential privacy if for all neighbor datasets D,D' $\in \mathscr{N}^{|\mathscr{T}|}$, $\|D - D'\|_1 \leq 1$, and all ranges $\mathscr{O} \subseteq \rho(\mathscr{R})$:*

$$Pr[\mathscr{M}(D) \in \mathscr{O}] \leq exp(\varepsilon) Pr[\mathscr{M}(D') \in \mathscr{O}] + \delta$$

If $\delta = 0$, the mechanism satisfies ε-differential privacy.

Intuitively, δ determines the probability of states that the difference between outputs on two neighbor datasets is more than factor of $exp(\varepsilon)$. If so, we will have a privacy breach. As the δ is smaller, differential privacy provides more confidence in privacy. Thus, setting δ to zero or having ε-differential privacy means we can trust the differential privacy mechanism that is resilient to any dataset and any auxiliary information.

Beside δ that specifies the robustness of differential privacy, ε is the key parameter in determining the power of differential privacy. For small values of ε we will have $e^{\varepsilon} \approx 1 + \varepsilon$. Which states that the output range of the mechanism over the neighbor datasets is approximately the same,

and an adversary cannot learn from the outputs. The choice of the value of ε determines the level of privacy that we expect from the differential privacy mechanism.

A higher ε value allows a looser bound on possible ranges of the output. In contrast, a lower value imposes more privacy restrictions and then, a tighter bound on the output ranges. A data curator is indeed interested in lower values of ε to assure a stronger privacy guarantee. On the other hand, a data analyst prefers higher values of ε, which allows more accurate outputs. Typically, ε is the *privacy budget* and a mechanism has to use it carefully to satisfy the requirements of both sides, privacy for the curator and accuracy for the analyst. The ε also determines the number of queries that can be answered before destroying or shutting down the database.

2.3.2 Queries and Sensitivity

As mentioned earlier, differential privacy controls the accessing mechanisms to a dataset. Each mechanism typically consists of a set of queries. Therefore, differential privacy relies on the type of queries, a key factor in determining the perturbation amount of the data. In this work, we are interested in aggregate queries that count the number of records satisfying some *predicates*. A *predicate* is a statement or condition that either holds or does not hold. This is formalized as $p : r \rightarrow \{0,1\}$ where $r \in \mathscr{T}$ and $\{0,1\}$ denotes whether the record satisfies the predicate or not.

Given a predicate p, we describe a *counting query* as a function $q_p : D \rightarrow \mathscr{R}$ where $D \in \mathscr{D}$ and $\mathscr{R}$ is the number of records in D that satisfy the predicate p. In the vector representation of a dataset D, we have non-negative integer values that indicate the count of records and a *counting query* is defined as follows.

Definition 2.4. *(Counting Query) Given a predicate $p : r \rightarrow \{0,1\}$ and a vector $v \in \mathbb{N}^{|D|}$ on a dataset $D \in \mathscr{D}$, a counting query q which satisfies the predicate p is defined as:*

$$q_p : v \rightarrow \mathbb{N} \cup \{0\}$$

where $\mathbb{N}$ is the set of natural numbers.

Counting queries are very powerful and are the main component of many data mining techniques. Let D be a dataset of features Age and Grade for students in a subject. After bucketizing, the vectorized dataset v contains $\{0,1\}$ values, 1 if a particular (Age, Grade) is in D and 0 oth-

erwise. A predicate p can be whether the Grade of a record is larger than a given value. A count query q_p is the number of records in v that satisfy the predicate p.

Generally speaking, an aggregate query defines a vector of $P(v) = [p(r_1), p(r_2), ..., p(r_{|v|})]$ based on the predicate p that assignes a weight to each record $r \in v$, and then, computes the inner product of these two vectors. So, the answer of an aggregating query computes as $q_p(v) = \sum_{r \in v} p(r)v(r)$. For a *counting query* the weighting vector consists of just $0s$ and $1s$.

Counting queries on ordered domains are mostly interested in a range of records that satisfy specific conditions. Indeed, the weighting vector in these queries contains a continuous range of $1s$ that specifies a consecutive set of records. We refer to this family of queries as *range queries* that apply on each dimension of the dataset. This query computes the total count of those objects that contributed to the query area, for example, the total number of cars passing through a road segment covered by the query area, or the number of smartphone users in each block of a city.

Intuitively, the weight of records are not restricted to $\{0,1\}$ and can be any real number. This concept generalizes the definition of *counting queries* and *range queries*. In this case, a query can be seen as a linear combination of the weight vector and the data vector, which is called *linear query*.

Definition 2.5. *(Linear Query) Given a predicate $p : r \to \mathscr{R}$ and a vector $v \in \mathbb{N}^{|D|}$ on a dataset $D \in \mathscr{D}$, a linear query q which satisfies the predicate p is defined as:*

$$q_p : v \to \mathscr{R}$$

where $\mathscr{R}$ is the range of output and a set of real numbers.

Without loss the generality, we assume the weights in $[0,1]$ and define a *linear query* as $q_p : v \to [0,1]$. Bounding the weights just simplifies the problem. It could also be used to compute the sensitivity of a set of queries from a specific family (will be discussed later). We will think of such set of queries as a *class of query* and formalize it as $Q : \mathscr{D} \to \mathscr{R}^t$ where t is the size of Q. A *class of query* could also be seen as a function that maps a dataset $D \in \mathscr{D}$ to a vector of real numbers $v \in \mathscr{R}^t$.

Going back to the definition of differential privacy, it adds an amount of noise to the query answer to hide the differences between each pair of neighbor datasets. Two key parameters identify

the amount of this perturbation: *privacy budget* (ε) and *sensitivity of the query class* denoted by Δ_Q.

Sensitivity of a query is the maximum difference of answers when the query is applied on neighbor datasets D and D'.

Definition 2.6. *(Sensitivity) For all pairs of neighbor datasets $D, D' \in \mathscr{D}$, the sensitivity of a query $q : D \to \mathscr{R}$ would be:*

$$\Delta_q = max\|q(D) - q(D')\|_1$$

A *counting query* has a sensitivity of one, $\Delta_q = 1$. Consider a counting query that is interested in *the number of students with grade $\geq$ 80.* Removing the record of one student would change the answer by 1, and each neighbor dataset withhold this change. For a *range query* that is a special type of count query, the sensitivity is also one. Therefore, a naive way to compute the sensitivity of a *class of counting queries* is to sum the sensitivity of included queries, $\Delta_Q = \sum_{q \in Q} \Delta_q$.

The sensitivity in *linear queries*, however, could be less than one. Since the weight of each record is at most 1, then removing one record might change the answer of a query by less than one vale. Accordingly, the sensitivity of a *class of linear queries* would be at most one, $\Delta_Q \leq 1$.

There are other classes of query with higher sensitivity. In these cases, more noise will be added to the answer of queries, and in turn, the published data would have less accuracy and usefulness. That is differential privacy works on queries with low sensitivities (at most 2). For queries with higher sensitivity, specific strategies should be applied to achieve privacy requirements of differential privacy.

2.3.3 Modes of Releasing Data

The problem of releasing private data in differential privacy view interprets as answering a set of queries. In other words, a *data analyst* or an *adversary* asks a set of queries $Q = \{q_i \in Q | q_i : D \to \mathscr{R}\}$. The *data curator* in turn computes the answer of queries and adds noise to the answers using a differential privacy mechanism. Also, these answers could be used to generate data that represents the original one.

The noise that the data curator adds should satisfy three requirements. First, it should comply with the privacy requirements of differential privacy to hide sensitive information. Second, the

released data should have adequate *accuracy* for further analysis. Definition of this notion depends on the *mode* of differential privacy in releasing data.

Another condition that the perturbed output should hold is preserving *usefulness* of the original data. Usefulness or *utility* can be considered as *consistency* of outputs. Suppose two counting queries q_1 and q_2 so that $q_1(D) \geq q_2(D)$ over original data D. A mechanism to satisfy consistency should preserve this relation while adding noise. Other definitions for *utility* of released data will be proposed and evaluated in this work. Chapter 4 will explain this notion in more detail.

Interactive Mode

The class of query Q may not be known for the data analyst. She/he may need to decide about the next query based on the answer of the previous one. Assuming that the data analyst will ask the query class $Q = \{q_1, q_2, ..., q_w\}$ queries, the data curator and the data analyst will have w interactions.

In each iteration i, the data analyst selects a query q_i using the answer a_{i-1} of the query q_{i-1}. Then, the data curator will compute the answer a_i for this iteration based on the previous answer a_{i-1}. Indeed, the mechanism in the *interaction mode accumulated* the noise in each iteration. This is because different releases of data may provide kind of *auxiliary information* for the data analyst to find out private information. Hence, accumulating the noise guarantees that the required bound for differential privacy will be held in each iteration.

However, accumulating the noise deteriorates the *accuracy* of output in subsequent queries. Accuracy of a query is usually defined in term of the error of query. We think of the error as the difference between the answer of the query on original data and its answer after perturbation, $error_{q_i} = \|q_i(D) - a_i\|_1$. Accordingly, the accuracy of a differential privacy mechanism in the *interactive mode* will be the sum of query errors. To guarantee the usefulness of the answer of all queries in the class Q, usually a parameter λ is used to bound the maximum accumulative error of the mechanism.

Definition 2.7. *(Accuracy in Interactive Mode) Given a query class $Q = \{q_i \in Q | q_i : D \to \mathscr{R}\}$ and a dataset $D \in \mathscr{D}$, a differentially private mechanism $M : D \to \mathscr{R}$ is λ-accurate if we have:*

$$Error_Q = \sum_{q \in Q} \|q(D) - M(D)\|_1 \leq \lambda,$$

where $\lambda > 0$.

Non-Interactive Mode

In *non-interactive mode* the query class Q is known, the data curator has the knowledge of all queries to compute and perturb the answers. The differentially private mechanism can exploit properties of all queries to generate the most accurate outputs. Depending on the output type of the mechanism, we will think of this case in two settings.

Sometimes the problem is just finding the answers to queries. So, the data curator computes the perturbed answers and release them. In the other setting, we need to release the whole dataset. In this case, the mechanism will use the noisy answer of queries to compute a similar distribution of the original data. We call this distribution as *Synthetic Data* $S \in \mathscr{D}$. The synthetic data should preserve the properties of the original data in answering the queries. The accuracy for this setting will then be formulated as the maximum error of queries over the synthetic data.

Definition 2.8. *(Accuracy in Non-Interactive Mode) Given a query class* $Q = \{q_i \in Q | q_i : D \to \mathscr{R}\}$ *and a dataset* $D \in \mathscr{D}$*, a differentially private mechanism* $M : D \to \mathscr{R}$ *is* λ*-accurate if the generated synthetic data* $S \in \mathscr{D}$ *satisfies:*

$$Error_Q = \sum_{q \in Q} \|q(D) - q(S)\|_1 \leq \lambda,$$

where $\lambda > 0$.

In this mode, we do not have the accumulation problem of noise for accuracy. However, the mechanism should be able to compute accurate and useful synthetic data for further analysis. We will discuss this setting in more details in chapter 3.

2.3.4 Composition Theorems

For interactive mode, we talked about adding *additive* noise for subsequent queries. In fact, the *interactive mode* is an instance of repeated use of a mechanism. In a series of analysis, a mechanism can be used just for a *single query* (applying a query multiple time), or for a *class of queries* (applying different queries). The point is that how we can accumulate the noise to assure the privacy

for each output. To simplify the problem, we focus on ε-differential privacy. The results could be extended to the (ε, δ)-differential privacy definition.

Referring to the definition of ε-differential privacy, ε is one of the parameters that determines the magnitude of noise. In the *sequential* setting the mechanism is applied several times on a the same dataset and we will have ε_i-differential privacy for in subsequent iteration. According to the following theorem, the total privacy level of mechanism for this sequence will be the sum of privacy budgets, $\varepsilon_i s$.

Theorem 2.1. *(Sequential Composition) [41] Given a mechanism $M_i : D \to \mathscr{R}$ that provides ε_i-differential privacy, the sequence r of $M_i(D)s$ provides $\sum_{i \in r} \varepsilon_i$-differential privacy.*

The sequential composition describes the privacy bound when the queries are dependent. When the queries are applied on disjoint subsets of the dataset, we do not need to accumulate the noise. Since the queries are independent in this scenario, we can assume that the dataset has been partitioned into disjoint subsets, and each query is applied to one of these subsets. In turn, the mechanism can be applied in parallel over all queries and the privacy guarantee of analyzing the query class would be identified by the worst privacy guarantee in queries.

Theorem 2.2. *(Parallel Composition) [41] Given a mechanism $M : D \to \mathscr{R}$ that provides ε-differential privacy and subsets $D_1, D_2, ..., D_k : D_i \in D$, the sequence r of $M(D_i)s$ provides ε-differential privacy.*

Disjoint queries are also called *histogram queries* as the subsets subjected to each query do not have any overlap.

2.3.5 Basic Mechanisms

So far, we have introduced the building blocks of differential privacy, including queries and sensitivity of each query type, privacy budget ε and composition theory in different interaction modes. Another key component in differential privacy is the mechanism that brings the other components together to compute private answers of queries.

In this section, we will discuss two basic mechanisms to implement differential privacy, *Laplace Mechanism* and *Exponential Mechanism*. Laplace mechanism assumes that the query

answers are in *numerical* and perturb them to preserve privacy. But, the exponential mechanism works for *categorical* outputs and allows to select an answer while providing differential privacy.

Laplace Mechanism

As we have seen, each query $q : D \to \mathscr{R}$ has a sensitivity of Δ. Also, we know that this sensitivity together with the noise magnitude ε identifies the scale of the random noise for hiding the difference between neighbor dataset.

Due to the properties of *Laplace distribution*, random numbers (noise) are generated from this distribution. Let $Lap(b)$ denote a Laplace distribution with density function

$$f(x|b) = \frac{1}{2b} exp\left(-\frac{|x|}{b}\right)$$

with *location parameter* $\mu = 0$ and *scaling factor* $b = 0$.

Now we are ready to define *Laplace mechanism* which perturbs each dimension of the data with a random noise from a Laplace distribution calibrated by *sensitivity* Δq of the query and the *noise magnitude* ε .

Definition 2.9. *(Laplace Mechanism) [40] Given a query $q : D \to \mathscr{R}$, a dataset $D \in \mathscr{D}$ and a privacy budget ε, a Laplace mechanism $LM : \mathscr{D} \to \rho(\mathscr{R})$ is defined as*

$$LM(D, q, \varepsilon) = q(D) + Lap(\frac{\Delta q}{\varepsilon}).$$

This definition can be extended to a *class of query* $Q : \mathscr{D} \to \mathscr{R}^t$. Then we will have $LM(D, Q, \varepsilon) = Q(D) + (l_1, ..., l_t)$ when l_i is the random variables with scale $\frac{\Delta Q}{\varepsilon}$ added to the dimension i.

Theorem 2.3. *The Laplace mechanism satisfies ε-differential privacy.*

As an example, consider the *count query q: How many students have higher grades than x in Math?*. The sensitivity of this query is one since adding or removing the record of one student with change the answer by one in the worst case. Then, by adding a random noise drawn from a Laplace distribution $Lap(\frac{1}{\varepsilon})$ we can provide ε-differential privacy for such queries. In this case,

the perturbation error is expected to be $\frac{1}{\varepsilon}$ (which is the expected value of absolute values for Laplace distribution).

Accordingly, for a *class of count queries* Q with size m depending on the structure of queries we can scale the noise to provide ε-differential privacy guarantees on the answer of queries. Following the discussion in the previous section, for independent queries, we must scale the noise to $\frac{m}{\varepsilon}$ and in histogram queries the noise improves to the scale of $\frac{1}{\varepsilon}$.

Another option for perturbing data is to add *Gaussian* noise. This distribution provides (ε, δ)-differential privacy which offers a weaker guarantee of privacy. The parameter $\delta \geq 0$ denotes the probability of leaking sensitive information.

Exponential Mechanism

The *exponential mechanism* is complementary for the Laplace mechanism. In some problems, we need to select the best choice among the range of outputs. The range of outputs may be nun-numeric which adding noise does not make sense. Or the outputs are numeric values but so sensitive that adding noise destroys them. In such a case, the exponential mechanism allows selecting the best output while preserving ε-differential privacy.

For example, suppose a classifier problem that we need to classify a set of labeled points $\mathscr{R}^2$ into two class of *Red* and *Blue*. Let the output l be a line that classifies the 2-dimensional space. In each iteration, the classifier computes the inner product of the line l and a point $r \in \mathscr{R}^2$. The point assigns to the *Red* class if the product is positive and otherwise assigns to *Blue*.

In this example, adding noise to the product may change the class of a point. Alternatively, as the goal is finding the best classifier l, adding noise to the classifier does not make sense. Informally speaking, the exponential mechanism for this problem assigns a usefulness score to each classifier and with high probability chooses the optimal one to classify the point.

Given a function $f : \mathscr{D} \to \mathscr{R}$ that generates nun-numeric range of $\mathscr{R}$, a score function $s : \mathscr{D} \times \mathscr{R} \to \mathbb{R}$ maps this output to a set of real values. For each dataset $D \in \mathscr{D}$, the probability of choosing output $r \in \mathscr{R}$ computes according to the score $s(D,r)$, the privacy budget ε and the sensitivity of the score function s.

Adding or removing a record from the dataset would affect the assigned score to a specific output r. Formally, the sensitivity of a score function $s : \mathscr{D} \times \mathscr{R} \to \mathbb{R}$ on all neighbor datasets D,

$D' \in \mathscr{D}$ and all possible outputs $r \in \mathscr{R}$ is defined as:

$$\Delta s = max_{r \in \mathscr{R}} max_{D,D' \in \mathscr{D}} |s(D,r) - s(D',r)|$$

Now we can define exponential mechanism as follows.

Definition 2.10. *(Exponential Mechanism) [42] Given a dataset $D \in \mathscr{D}$, a possible output $r \in \mathscr{R}$ ans a score function $s(D,r)$ the probability of selecting r as the output using the exponential mechanism $EM(D,s,\mathscr{R})$ is proportional to $exp(\frac{\varepsilon s(D,r)}{2\Delta s})$.*

Indeed, the intuition behind perturbing the choices proportional to $exp(\frac{\varepsilon s(D,r)}{\Delta s})$ is to give the most probability to the best choice and decrease the probability of inferior outputs exponentially. It has been shown that it is very unlikely in the exponential mechanism to choose an inferior output [42].

Note that the privacy budget ε in exponential mechanism has been divided by 2. By this division, the mechanism saves half of the privacy budget for effects of normalization term when adding or removing a record, decreases the score of some outputs and increases the others [22].

Theorem 2.4. *The exponential mechanism satisfies ε-differential privacy.*

2.4 Differentially Private Location Data Publishing

Standard mechanisms for preserving differential privacy (e.g. Laplace mechanism (LM) and exponential mechanisms (EM)) are basically proposed for the interactive version, where the adversary asks a query from data holder and receives a noisy answer. When it comes to the non-interactive version where the whole dataset should be released, these mechanisms are not practical due to accumulation of noise for all possible queries. To address this issue suggested algorithms exploit properties of data or the query class to improve the accuracy, which are then called data-aware and query-aware, respectively. In this section, we review recent data- or query-aware mechanism introduced for publishing *location dataset* and describe their pros and cons.

2.4.1 Data-Aware Algorithms for Location Datasets

The intuition behind data-aware algorithms is finding most informative dimensions in data and scaling the nose accordingly rather than scaling it considering all the data dimensions. Data-awareness enhances the accuracy of outputs by reducing the scale of added the noise. Rastogi et al. [43] introduced *SPA*, which uses discrete Fourier transform. The algorithm first computes Fourier coefficients and then employs random sampling to select k most informative coefficients. In the next step, *SPA* applies Laplace mechanism so that it satisfies differential privacy and obtains original data using inverse Fourier transform. Gergely et al. [44] proposed *EFPA* that improved accuracy of the SPA by enhancing EM in selecting the k coefficients and defining a more accurate utility function for weighting coefficients. *Privelet* [45] is another algorithm that exploits Haar wavelet transform and Laplace mechanism to build a faster and more efficient algorithm. The three algorithms consider the class of range/counting queries and use histograms of original data values as input to release noisy counts of histograms. The assumption is that the data owner has applied arbitrary range/count queries on the original location dataset to construct aggregated data as a histogram. The goal of the above algorithms is to publish this aggregated data under differential privacy. The models, however, fail to provide a tight error bound to guarantee the accuracy of outputs [46].

Subsequent data-based mechanisms improved the error bound of publishing aggregated data by using more information about the data. In particular, the mechanisms used clustering techniques to get different density partitions in the dataset before applying any data transformation. Intuitively, having clusters of similar counts, values in a cluster can be replaced by the mean value of that cluster, and hence, the mechanism needs to add noise just to the clusters central point. This strategy reduces the sensitivity of the mechanism and then perturbation of data. *P-HPartition* [44] is an algorithm based on this technique. It takes a list of raw histograms as input and iteratively generates a tree of histogram clusters. In each iteration, this algorithm utilizes exponential mechanism to choose the best cluster. P-HPartition is a modified version of two earlier methods *SecureFirst* and *NoiseFirst* [47]. SecureFirst computes clusters using exponential mechanism and then adds noise based on Laplace mechanism. In contrast, NoiseFirst adds noise to data using Laplace mechanism and then computes best partitions. Due to dependency on cost function to the data, SecureFirst and NoiseFirst are not scalable and on large datasets have low utility. Moreover, data

owner should identify the number of clusters as the input of the algorithms. P-HParttion resolves these drawbacks by automatically selecting the best clusters and introducing an effective cost function, which is independent of data and improves the accuracy of output and the algorithm's computations complexity.

2.4.2 Query-Aware Algorithms for Location Datasets

Despite the data-based algorithm, query-based model focus on optimizing the published data for answering a particular class of queries. Such mechanisms apply basic differential privacy mechanisms, including exponential mechanism and Laplace mechanism, to compute differentially private answers and then boost the accuracy of answers using post-processing techniques. Moreover, due to the random noise, the noisy answers may not be consistent with real ones. Consider an analyst who wants to know the number of students with s grade higher than (a) 80 and (b) 75 in Mathematics. Semantically the answer of former query, the number of students with a score higher than 80, should be less than the latter. However, naively adding random noise to the output using the basic mechanisms, we may have more students with grade 80 than those with 70!

To address the consistency as well as accuracy of outputs in a differentially private model, Hey et al. [48] assume the data owner knows the consistency constraints in queries and then, based on this assumption they propose two post-processing approaches named *BoostMM* and *BoostTree* to reduce the error of answers and guarantee the consistency of outputs. The approaches are developed for answering count queries on location datasets. They take a histogram as input and apply Laplace mechanism to add noisy to the data. BoostMM assumes histograms are ordered and uses least square regression for post-processing. Whereas, BoostTree works for dependant queries where changing one bin in the original histogram may affect the answer of several queries. Given the pre-defined constraints in queries, they apply constraint programming to release more accurate histogram counts and ensure the consistency as well. To guarantee accuracy and consistency, Barak et al. [49] employ Fourier transform and linear programming and release an approximation of data. In the first step, Laplace mechanism is used to generate a private intermediate output. Fourier transform in the second step reduces dimensions of data for the next step in which linear programming enhances accuracy while preserving the consistency in the generated outputs.

Chao et al. [50] show that each query-based model is an instance of a general mechanism called

Matrix Mechanism (MM) that defines a transformation matrix, namely query strategy, based on input queries to reduces the required noise and then improves the accuracy and consistency. This mechanism considers linear queries and clarifies why the prior works are not accurate and have large error bounds. According to MM, for each set of input queries, it is possible to find a query strategy, which has the lowest sensitivity and provides the same level of privacy comparing to other query strategies. In another view, the transformation matrix used in prior works has not had the lowest sensitivity, which caused low accuracy in outputs.

Even though the problem of releasing data differentially private depends on both data and query properties, most the works have considered only a part of the problem. As mentioned, data-based algorithms focus on the properties of data and ignore the features of input queries. On the other hand, query-based models improve the accuracy of the mechanism according to characteristics of the input query set. Chao et al. [51] describe an algorithm, *DAWA*, that considers properties of both data and input set of linear queries. DAWA releases noisy histograms in three steps. First, it finds the best clustering for input histograms so that buckets in each cluster are approximately uniform. In this step, the algorithm uses MM to make the partitioning private and then computes noisy values of buckets for the best partitions determined in step one. The third step expands the noisy value uniformly among buckets in that cluster. However, although this algorithm works efficiently on small datasets, it is not scalable to high dimensional data. Moreover, finding the query strategy with the lowest sensitivity in MM has a very high time complexity, mainly when the input query represents the whole class of query (means the released data should be useful for any query in that class not just a subset of possible queries).

Recently, Moritz et al. [46] introduced an algorithm called Multiplicative Weights Exponential Mechanism (*MWEM*) that aims to enhance the running time for releasing synthetic data differentially private and useful for a whole class of linear queries. To this end, MWEM received the number of most informative queries (T). Then, after computing noisy answers using Laplace mechanism, the algorithm chooses the most informative query based on exponential mechanism and improves the approximation over original data. Selecting small values for T and parallelising the computations, the algorithm ends in few seconds and improves both accuracy and running time of prior works for an arbitrary number of queries [52] [53] but still suffers from low accuracy.

Although MWEM has the best worst-case error bound compared to other described algorithms,

it has several drawbacks. First, it works well only on datasets that are highly uniform and has a poor performance on datasets with many dense areas. Second, the algorithm time complexity depends on the number of iterations, T. High dimensional datasets need higher T, and then the running time grows exponentially. Third, the Multiplicative Weights technique is highly sensitive to high errors and cannot adjust the scaling factor in the worst cases. This problem leads to poor approximation and then the low utility of released data.

Tables 2.1 summarizes the above mechanisms for publishing location datasets and contrast their technical aspects.

Name	Query Type	Basic Mechanism	Data/Query Aware	Time Complexity	Error Rate	Notation	Reference
Privelet	Count	LM	Data Aware	$O(m+n)$	$O(log\frac{\|D\|}{\sigma^2})$	n: # records; m: # distinct values in attribute domain; σ: noise variance	[45]
SPA	Count	LM	Data Aware	$O(nlogn)$	$\sqrt{\sum_{i=k+1}^{n}\|F_{i-1}\|^2}+\frac{k\sqrt{n}}{\varepsilon}$	k: Fourier coefficient	[43]
EPFA	Count	EM	Data Aware	$O(nlogn)$	$\sqrt{\sum_{i=k+1}^{n}\|F_{i-1}\|^2}+\frac{2k}{\varepsilon}$	n: # coefficients	[44]
PH-Partition	Count	EM	Data Aware	$O(n^2)$	$O(logn)$	n: # buckets	[44]
Secure First	Count	LM, EM	Data Aware	$O(kn^2)$	$k(F+1)$	F: max # bins; n: # records; k: # bins	[47]
Noise First	Count	LM	Data Aware	$O(kn^2)$	$SSE_D+\frac{2k}{\varepsilon}$	SSE_D: sum of squared error over D; n: # records; k: # bins	[47]
BoostMM	Count	LM	Query Aware	$O(n)$	$O\left(\frac{dlog^3n}{\varepsilon^2}\right)$	d: # distinct values in bins	[48]
Boost Tree	Count	LM	Query Aware	$O(n)$	$O(\frac{l^3}{\varepsilon^3})$	n: # bins; l: level in tree of queries	[48]
DAWA	Count	MM	Data and Query Aware	$O(n^2logn)$	$O(n^2+nlog(\frac{\|B\|}{\delta})/\varepsilon)$	n: # buckets; B: set of all intervals	[51]
MWEM	Linear	LM, EM	Query Aware	$O(n\|Q\|+T\|D\|\|Q\|)$	$O(n\sqrt{nlogQ})$	n: # records; Q: # queries	[46]

Table 2.1: Contrasting current algorithms for publishing location datasets as a simple histogram.

2.5 Differentially Private Trajectory Publishing

Considering each location as a dimension in a continuous spatial domain, a trajectory is a high-dimensional object consisting of a *sequence* of locations. The sequential dependency of locations in a trajectory entails an unbounded sensitivity in a naive differential privacy mechanism. Moreover, the aforementioned algorithms for publishing location data do not consider this dependency among locations which makes the algorithms impractical for trajectory datasets. For example, given the trajectory of a user traveling along a coastal road, a naive randomization algorithm might perturb the GPS points in such a way that most of the user's locations are in the sea, leading to little utility. Also, conventional space aggregation and partitioning methods are not applicable in trajectory publishing mechanisms due to this sequential dependency in data.

The mechanism for publishing trajectory datasets should overcome the sequential dependency scaling the added noise, which directly identifies the utility of published data. To address this challenge, existing works take one of these strategies: (1) Aggregating trajectories as a histogram of counts computed by *range queries* and adding noise to the aggregated counts, (2) Mapping trajectories into a data structure that captures accurate statistics over movement patterns in trajectories and then, sampling synthetic trajectories under differential privacy. In this section, we describe the proposed approaches for each strategy and explain their pros and cons.

Note that in this thesis we target privately publishing of trajectories given a trajectory dataset. Another research direction is privately publishing trajectories in real-time which is beyond the scope of this thesis. For example, in commercial applications, it is required to know whereabouts of individuals at regular time intervals to provide targeted advertisements. Proposed mechanisms in such real-time scenarios (e.g., [54], [55]) ensure the privacy of user by providing differential privacy guarantees when the data is generated.

2.5.1 Publishing Aggregated Counts Over Trajectories

Aggregating trajectories based on a particular query type is inspired by mechanisms developed for publishing location data under differential privacy (see Section 2.4). However, answering queries on trajectories has challenges inherited from the sequential dependency of location point in a trajectory which in turn causes a dependency between components of aggregated data. Given a

spatial domain and a temporal domain wherein the trajectories are defined, an aggregated data means coarsening these continuous domains to distinct domains such as a grid, and counting the number of *distinct* trajectories in the constructed regions of coarsening space, i.e., *range query* on trajectories.

Answering range queries on trajectories has received a considerable attention in spatial data management and processing [12], [56], [13], [14]. Due to the large volume of data, the data structure should (1) efficiently store the data, (2) efficiently process the query, and (3) report a correct answer. Flajolet and Martin [56] devised a probabilistic approach using hash functions that is memory efficient and uses trajectory IDs to aggregate information into a bitmap. However, this approach and further probabilistic methods based on that (e.g., [11]) may entail a significant error in approximating the correct count for a range query. Moreover, preserving the trajectory ID threatens the privacy of the individuals. Studies [57], [58] show that human trajectories are often unique and that a trajectory ID can be considered as the individual's identifier. Xie et al. [14] designed a spatial histogram by decomposing the space into cells and aggregating the count of trajectories overlapping with each cell. This so-called Distributed Euler histogram (DEH), uses properties of Euler formula [59] to decompose the space to count the number of static objects. DEH is very memory efficient, can exactly compute range queries without using trajectory IDs. It efficiently computes range queries with $O(k)$ computational cost where k is the number of cells in the query range (see Spatial Histogram in Section 3.2). Subsequent researches [60], [15] introduced variants of this histogram for particular types of trajectory movements, while the basic structure is the histogram introduced by [14]. Leonardi et a. [13] developed a spatial histogram that computes the exact answer for all trajectory movement types. This histogram has a similar structure as DEH except that it needs trajectory IDs to compute the query answer, which in turn imposes a privacy threat.

Even though a spatial histogram provides aggregated information of trajectories in the given space, it does not guarantee the privacy of individuals therein. Montjoye et al. [57] showed that coarsening does not significantly reduce the uniqueness of trajectories and individuals are susceptible to privacy attack in the aggregated data. The threat is stronger when the data is sparse, or an area has a low density of trajectories. Even though this shows the difficulty of achieving a privacy guarantee using k-anonymity (aggregating k trajectories with same sequence of location points)

there are many attempts in this regard (e.g., [58], [61], [62]). A recent study [5] showed that given aggregated data of individuals' trajectory, anonymized by k-anonymity approaches, a scale of tens of thousands to hundreds of thousands of trajectories could be accurately recovered.

Thus, to ensure strong privacy guarantees, our goal is to develop a differentially private mechanism for synthesizing spatial histograms, which provides aggregated information of trajectories and efficiently answers spatial range queries. Multiple prior works [18], [20] construct a hierarchical structure using the given dataset and synthesize aggregated information of trajectories while satisfying differential privacy. Chen et al. [21] use a Markov process to model local sequentiality and construct a hierarchical structure such as a prefix tree by grouping the trajectories with the same prefix. Subsequent works [19], [20] use most frequent substrings to get higher counts in the leaves and to achieve a better utility. However, a recent work [63] showed that the mechanisms fail to preserve trajectory properties such as their length, and thus cannot guarantee the accurate distribution of trajectories. Additionally, the approaches are not efficient in computing range queries. A range query specifies an *area* which is a continuous spatial region and every trajectory passing through this area should be counted. Existing works publish a synthetic dataset of trajectories that means search techniques such as hierarchical structures should be applied on the synthetic trajectories to compute the number of distinct ones in the query area. Given the continuous spatial space of data, a hierarchical structure divides the space into *independent* areas in which each area represents a node in the hierarchical structure. The area specified by a range query may overlap multiple nodes from different branches of the hierarchical structure. For finding the distinct number of trajectories in a query area, the algorithm may need to search all branches of the hierarchical structure to (1) find whether a node (partially) maps to the query area and (2) compute the total number of distinct trajectories crossing the node. Considering the continuous data space and the size of query area, the total number of nodes to be searched in the hierarchical structure may be a large value. For example, let the data space represent a metropolitan area of size $(10 \times 10)km^2$ and the queries in query set Q target areas of minimum $(1 \times 1)m^2$ size. In this case, the data space should be divides into 10^8 areas of size $1m^2$ to precisely compute the number of trajectories for even the smallest query area. Constructing a quad-tree to search the space will requires a tree of height $h = 13$ with branch factor $a = 4$ where all a^{h+1} nodes should be searched in worst case when a large query overlaps multiple nodes in different branches. Hence, comput-

ing a range query using conventional hierarchical structure with height h and branching factor a has $O(a^{h+1})$ computational cost, in contrast to a spatial histogram with $O(k)$ simple operations of addition and subtraction, where k is the size of query range range, i.e., the number of distinct areas overlapping the query. Other than computational complexity, trajectories in contrast to locations may cross multiple nodes in a hierarchical structure. This feature in trajectories' movement makes conventional search approaches impractical in counting the number of trajectories as they assume the nodes are independent. Consider an example range query counting the number of distinct vehicles passed through an area containing suburbs S1, S2, S3. Given a hierarchical structure considering each suburb as a node in different branches, the search technique has to go through different branches of the structure to find the suburbs and count the trajectories. Other than this computational inefficiency, the computation strategy should ensure total trajectories counted in the three suburbs represent distinct ones. If a trajectory has visited all the three suburbs the hierarchical structure does not provide any information about the dependency of these suburbs and hence, it is not possible to ensure whether the counted trajectories are distinct.

Monreale et al. [64] define the dependency between cells instead of points by mapping the trajectories to a grid to count the movement frequencies between the adjacent cells. However, a frequency vector only maintains the number of transitions for a group of observations without information about the spatial adjacency of two vectors. This information is crucial for a range query as it counts the vectors overlapping the query area. In our work, we use a spatial histogram that is a grid but captures both the cell counts and the spatial adjacency of trajectories.

Related work in coarsening the data space are in two categories: data-aware mechanisms [65], [66] and query-aware mechanisms [7], [46, 51]. The data-aware mechanisms use some data properties to adaptively partition the data space and optimize the error of published data in answering range queries. However, the models are developed for location points rather than trajectories. Cormode et al. [65] build a hybrid tree in which a kd-tree partitions the space into coarse density regions and a quadtree maps the dense areas into finer resolutions. Qardaji et al. [66], generate partitions to get uniform regions of data. The partitioning condition is based on the density of location points in regions. The data-aware partitioning techniques make use of information about the data distribution to enhance accuracy. However, the mechanisms are not able to capture local sequentiality of location points which is crucial in representing trajectories. The query-aware

mechanisms, on the other hand, take a set of range queries to get information about the data distribution. Laplace mechanism [7] is a naive query-aware model that adds an independent noise to the answer of each query in the query set. Hardt et al. [46] developed MWEM that optimizes the accuracy of published data for the class of range queries. By laying a grid with a fixed cell size on the data space, MWEM iteratively learns the density of cells using the noisy answer of queries. Li et al. [51] proposed a data- and query-aware (DAWA) which employs data properties as well as query properties in publishing data under differential privacy. A comparative analysis of space decomposition mechanisms by Hay et al. [67] showed MWEM and DAWA achieve high accuracy in answering range queries on location data, not trajectories. Since the mechanisms are not able to capture the local dependency between locations which is crucial in studying trajectories.

2.5.2 Publishing Trajectory Datasets

Achieving a trade-off between privacy and utility requires this category of mechanisms to overcome two main challenges: *high-dimensionality* and *sequentiality* of trajectories. Considering each location as a dimension in a continuous spatial domain, a trajectory is a high-dimensional object with an arbitrary length, i.e., the number of locations in trajectory. The unlimited length of trajectory entails an unbounded scale of noise in a naive mechanism. Also, the locations in a trajectory are sequentially dependent which makes the conventional space aggregation and partitioning methods [65] inapplicable in trajectory publishing mechanisms.

Chen et al. [21] utilized the sequential dependency to fit the trajectories to a Markov model and developed a noisy prefix tree such that synthetic trajectories could be generated. Partitioning the data into disjoint groups reduces the possible movement options (i.e., size of the spatial domain) and hence, the magnitude of noise to ensure differential privacy. Subsequent works [18], [19], [20] extended the idea of aggregating trajectory counts in a hierarchical structure. Chen et al. [18] treated a trajectory as a string and extracted n-grams to improve the utility of synthesized trajectories. Later, Bonomi et al. [19] reduced the error of constraining trajectories to their frequent patterns by making use of a transformation technique. Recently, He et al. [20] considered trajectories as spatio-temporal objects and developed a mechanism to capture the temporal resolution changes while generating synthetic trajectories.

He et al. [20] used multi-level discretization of the spatial domain to obtain velocity changes,

however, stay locations (i.e., the moving object has stayed in the location for some continuous time intervals) as key elements in semantic analysis of trajectories yet to be addressed. Jiang et al. [68] investigated the trajectory properties to add noise to a trajectory rather than their aggregation directly. They made use of sampling and interpolation methods to construct a trajectory privately. However, the mechanism assumes the trajectory length is known, and the starting and ending points are not sensitive to be protected.

Among trajectory publishing approaches, mechanisms based on Hierarchical structures are able to capture the sequentiality in trajectories by utilizing Markov models but fail to preserve the other inherent features of a trajectory dataset. First, the mechanisms assume that (significant information of) trajectories can be represented by a few location points. Limiting a trajectory to its first few [20], [21] or top frequent [18], [19] locations cannot represent the trajectory movement pattern. In particular, constraining the *length* of trajectories may severely degrade the utility of data. Second, due to the large size of the spatial and temporal domain, the mechanisms need to *prune* the hierarchical structure to improve the signal to noise ratio and hence the utility in the leaves. Removing some extra locations from truncated trajectories imposes an extra constraint on the utility of resulting data *distribution*. Finally, the employed structure is not able to encode the *temporal* feature of trajectories which provides key information about the *semantic* of trajectories.

The proposed structures are not able to capture the stay locations information and merely consider a single point for each visited location. Analyzing stay locations is the main research topic in spatial and spatial-temporal data mining with the objective of understanding interest regions of users [69]. Huo et al. [70] proposed an algorithm for preserving stay locations by splitting trajectories based on their stay locations and then aggregating the segments to ensure privacy. However, the algorithm cannot provide a strong guarantee of privacy. Ho et al. [71] applied aggregation and space decomposition techniques in order to identify stay locations differentially private but is not scalable for publishing trajectories.

It is imperative for the utility of generated trajectories to fit the dataset onto a probabilistic model such that the model carefully captures the inherent spatial and temporal features of trajectories with *no underlying assumption about their length or movement patterns*. This motivates us to propose our mechanism for differentially private publishing of trajectories while preserving the semantic of trajectories.

Input Type	Output Type	Query Type	Basic Mechanism	Intuition
Location	Aggregated (simple histogram)	Count/Range	LM, EM	Partitioning the spatial space using Quad-tree [65], kd-tree [65], uniform or adaptive grid [66] and adding noise to the location counts in each partition
Trajectory	Aggregated (spatial histogram)	Range	LM, EM	Coarsening the spatio-temporal space using a grid [14] [60] [15], applying a probabilistic model for randomizing the counts in grid cells (PriSH in Chapter 3); Quad-tree for uniform partitioning of counts and randomizing the partitions counts (DQAM in Chapter 4)
Trajectory	Trajectory	Any	LM, EM	Prefix-tree [20], [21] [18], [19] to capturing movements; Graphical model to capture movements and guided sampling to generate trajectories (TGM chapter 5)

Table 2.2: Taxonomy of differentially private mechanisms for publishing spatio-temporal data.

2.6 Conclusion

In this chapter, we review well-known privacy models and described their pros and cons. As it was discussed, privacy models before differential privacy (introduced as syntactic models) fail to provide a proven privacy guarantee. In this thesis, we focus on differential privacy model to develop algorithms for publishing trajectory datasets. We described trajectories in the chapter and discussed the challenges of publishing such spatio-temporal data under differential privacy.

Table 2.2 summarized approaches proposed for publishing spatio-temporal datasets under differential privacy in three categories: (1) aggregated data over locations, (2) aggregated data over trajectories, and (3) synthetic trajectories generated based on original trajectory dataset. In this thesis, our focus is on publishing trajectories, and we propose mechanisms for publishing both aggregated or synthetic trajectories. By adapting conventional hierarchical models to the trajectories and employing probabilistic models, we introduce new approaches for publishing aggregated counts. We also propose a new structure that accurately captures statistics of movements in trajectories and then we employ our structure to propose a novel mechanism for publishing synthetic trajectories with significant utility in contrast to the existing approaches.

Chapter 3

Publishing Spatial Histograms Under Differential Privacy

Studying trajectories of individuals has received growing interest. The aggregated movement behaviour of people provides important insights about their habits, interests, and lifestyles. Understanding and utilizing trajectory data is a crucial part of many applications such as location based services, urban planning, and traffic monitoring systems. Spatial histograms and spatial range queries (see Section 2.3.2) are key components in such applications to efficiently store and answer queries on trajectory data. A spatial histogram maintains the sequentiality of location points in a trajectory by a strong sequential dependency among histogram cells. This dependency is an essential property in answering spatial range queries. However, as discussed in Chapter 2, the trajectories of individuals are unique and even aggregating them in spatial histograms cannot completely ensure an individual's privacy. A key technique to ensure privacy for data publishing is ε-differential privacy as it provides a strong guarantee on an individual's provided data. In this chapter, we propose a mechanism that for the first time guarantees ε-differential privacy for spatial histograms on trajectories, while ensuring the sequentiality of trajectory data, i.e., its consistency. Consistency is key for any database and our proposed mechanism, PriSH, synthesizes a spatial histogram and ensures the consistency of published histogram with respect to the strong dependency constraint. In extensive experiments on real and synthetic datasets, we show that (1) PriSH is highly scalable with the dataset size and granularity of the space decomposition, (2) the distribution of aggregate trajectory information in the synthesized histogram accurately preserves the distribution of original histogram, and (3) the output has high accuracy in answering arbitrary spatial range queries.

3.1 Introduction

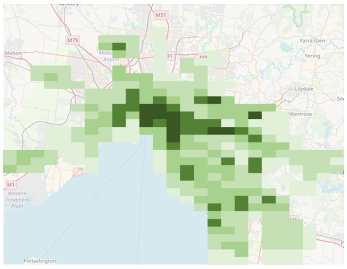
DUE to popularity and ubiquity of GPS-enabled devices (e.g., wearables, smartphones and vehicles), there are enormous amounts of trajectory data capturing moving behaviour of individuals. These large trajectory datasets are needed in a wide range of applications such as traffic management services (e.g., congestion detection and control), urban planning (e.g., facility setup, resource management in mobile communication and transport planning) and commercial purposes (e.g., identifying hotspots for advertisement).

A key query in understanding trajectory data and offering location-based services is the *spatial range query*[1]. This query computes the total count of those trajectories that contributed to the query area, for example, the total number of cars passing through a road segment covered by the query area, or the number of smartphone users in each block of a city. The spatial range query is also known as the aggregate query or distinct count query and has been extensively studied in the (spatial) data management literature [11] [12] [13].

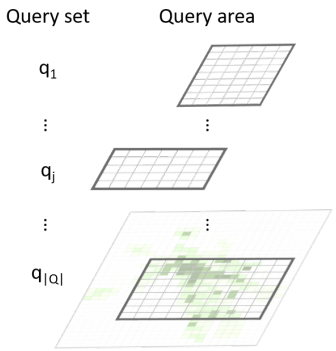
To correctly answer spatial range queries, *spatial histograms* are popular data structures [14] [15] that efficiently store trajectory data. A spatial histogram decomposes the underlying space into finite *cells*. Each cell captures the number of trajectories intersecting with the cell. In contrast to *simple histograms* where cell counts are independent, in spatial histograms, there is a strong dependency between the cell counts. A trajectory may overlap a sequence of cells and thus, adding/removing the trajectory leads to changing the values of all contributing cells. In a spatial histogram, changing one cell value may change a sequence of adjacent cells. This sequentiality constraint is the key property of spatial histograms that enables them to answer range queries on trajectories. We use Distributed Euler Histograms (DEHs) [14] as spatial histograms as they can count the *unique* number of trajectories by keeping trajectory counts for each cell as well as counts for each trajectory intersections of edges between two adjacent cells.

Despite the importance of spatial histograms in research and commercial applications, data providers need to take privacy concerns into consideration before publicly releasing data. Even though a histogram aggregates trajectory data, an individual's identity might still be inferred. Xu et al. [5] showed that up to 91% of trajectories could be uniquely recovered in aggregated movement trajectories. This problem is even more severe in rural areas with a low density of trajectories. However, anonymizing a spatial histogram is difficult, as changing a cell value, randomly and in-

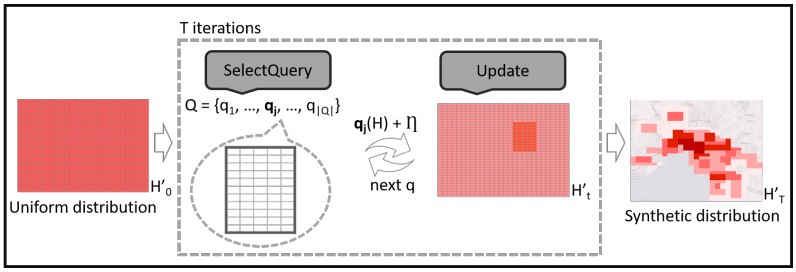
[1] Spatial range query and range query are used interchangeably throughout the chapter.



(a) Spatial histogram (H)



(b) A set of range queries (Q)



(c) Learning/synthesizing process

Figure 3.1: PriSH mechanism overview: H and Q in (a) and (b) are inputs of the learning process in (c) to synthesize the histogram.

dependent of its dependency with adjacent cells, may violate the consistency of values and hence, compromise the utility of anonymized histogram.

A robust and strong privacy model, such as differential privacy [7] is needed to guarantee the privacy of a published histogram. Besides, the anonymization mechanism needs to ensure the consistency of published histogram regarding the cell dependency and high accuracy in answering range queries.

Our Contributions. We propose a *Private Spatial Histogram* (PriSH) mechanism that outputs a synthetic spatial histogram with differential privacy guarantees. To the best of our knowledge, we are the first to address the problem of privately publishing spatial histograms. In the synthesis process, PriSH utilizes the data dependency and ensures the consistency of cell values in the published histogram. This process exploits range queries in optimizing the accuracy of the generated histogram and ensures high accuracy in answering spatial range queries. We summarize our contributions:

1. We propose PriSH which publishes spatial histograms with differential privacy guarantees.

2. We develop a private mechanism that utilizes data properties in synthesizing the data and ensures the consistency of cell values in the published histogram.
3. Our mechanism ensures high accuracy in answering spatial range queries.

Mechanism Overview. Our mechanism receives a spatial histogram and a set of range queries as input. Figure 3.1(a) shows a spatial histogram H created from walkers' trajectories in Melbourne city [72]. The values in H are displayed in colors (darker represents a more populated region). Figure 3.1(b) shows a set of random range queries Q with different query areas over the region where the queries may overlap. PriSH privately learns the distribution of H and generates a synthetic histogram H'. The learning process initially considers a uniform distribution H'_0 and iteratively improves the estimation toward the true distribution of H (Figure 3.1(c)). In each iteration, the mechanism privately selects the most informative query (*selectQuery* function) and updates the estimation by rescaling the values (*update* function) using the noisy answer of the query. The uniform distribution initially considers an equal weight for all values in shaping the true distribution. The selected query provides new information about a particular section of histogram H, i.e., the noisy answer of query. The mechanism updates that section by rescaling the weights with respect to the noisy answer. Note that due to differential privacy requirements, the learning process is limited to a few iterations to compute an accurate estimation. This constraint underlines the importance of designing the *selectQuery* and *update* functions in PriSH.

Recently, Hardt et al. [46] proposed an approach, called MWEM, for publishing *simple histograms* that works well on small/mid-size datasets with a simple distribution (i.e., few density regions in data), and can be applied on a continues spatial domain by discretizing the space into *independent* cells [67]. However, the independency assumption limits MWEM to study the distribution of location points rather than trajectories as there is no sequentiality constraint among locations. The update function in MWEM rescales the cell counts independently, which may significantly limit the utility of estimated distribution for spatial histograms.

This limitation is quantified in Figure 3.2. We considered the taxi trajectories in a large metropolitan area [73] covering a region about $20km \times 20km$ and mapped it to a spatial histogram. A small sample of 1000 trajectories was taken to create a sparse histogram — i.e., less information to the estimation process. We discretized the space with 8 resolution levels to capture the distribution of trajectories with different granularities. The size of spatial histogram therefore varies from

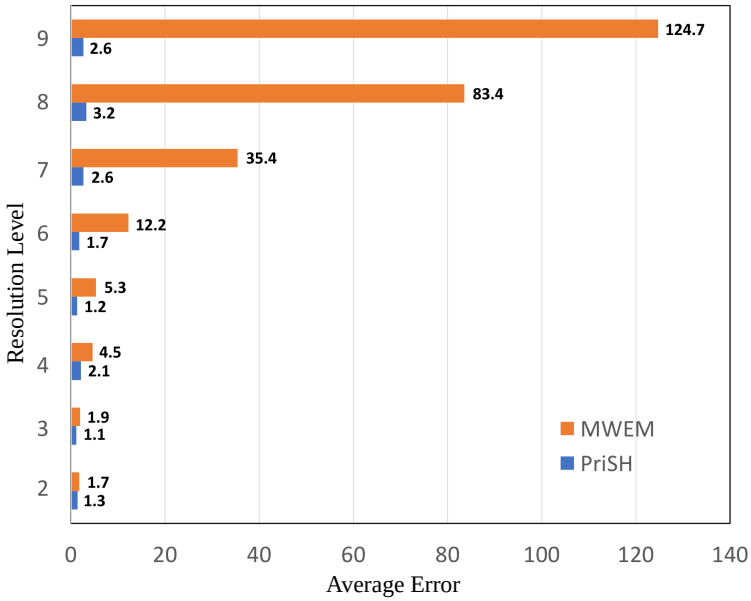


Figure 3.2: Accuracy of PriSH vs. MWEM in answering range queries.

$2^2 \times 2^2$ cells of $100Km^2$ to $2^9 \times 2^9$ cells of $19.5m^2$. We applied MWEM mechanism to synthesize the spatial histogram in each resolution. The generated histogram is then used to answer the given range queries Q, which was used in the learning process. We observe that the query error is exponentially increased in higher resolutions, which shows the inconsistency of synthesized histogram by MWEM in different resolutions.

Our proposed mechanism, Private Spatial Histogram (PriSH) is a scalable mechanism with a strong privacy guarantee. It can accurately synthesize spatial histograms from coarse to very fine granularities. PriSH ensures the consistency of generated histogram in answering range queries and bounds the query error to the precision of the learning process.

Organization. Settings considered for differential privacy are detailed in Section 3.2. Our mechanism and its main functions (*update* and *selectQuery*) are discussed in Section 3.3. The performance of mechanism is evaluated in Section 3.6, and Section 3.7 concludes the chapter.

3.2 Preliminaries

3.2.1 Dataset and Queries

Let $\mathscr{S} = \{s_1, s_2, ..., s_N\}$ be the spatial universe and N the size of universe. We approximate the continuous spatial universe with a discrete universe of N points. Each location point $s_i \in \mathscr{S}$ is specified by a pair of latitude, longitude degrees. Given a spatial universe $\mathscr{S}$, a trajectory t with length $|t| = l$ is an *ordered sequence* of l points drawn from $\mathscr{S}$ —, i.e., $t = \{s_{t_1} \rightarrow s_{t_2} \rightarrow ... \rightarrow s_{t_l} | s_{t_i} \in \mathscr{S}, s_{t_i}$ and $s_{t_{i+1}}$ are adjacent cells, $\forall i \in \{1, 2, ..., l\}\}$. A trajectory represents the history of locations a moving object has passed through.

A trajectory dataset consist of n records, each representing a unique trajectory. We simplify the notation and define a trajectory t of length l as $t \in_o \mathscr{S}^l$, where "$\in_o$" indicates the ordered instant and length $l \in \mathbb{N}$. Then, a trajectory dataset D with size $|D| = n$ is defined as

$$D = \{t_i | t_i \in_o \mathscr{S}^l, \forall i \in \{1, 2, ..., n\}\},$$

where a trajectory is in $\mathscr{S}^l$ if its length is at most l. The universe $\mathscr{U}$ of trajectories in D represents all trajectories of .

A *spatial range query* on a trajectory dataset reports the number of distinct trajectories in the query area. We write it as function q:

$$q : \mathscr{U} \rightarrow \mathbb{N}_0$$

which maps the trajectory domain $\mathscr{U}$ into a non-negative integer domain $\mathbb{N}_0$. In a range query, the contributed trajectories are not restricted to have common location points to be counted. This feature makes the class of range queries computationally expensive (with the order of $O(n \times N)$ for $n = |D|$ the size of trajectory dataset and $N = |\mathscr{S}|$ the size of spatial universe) on raw trajectory datasets as the intersection of each trajectory (i.e., a sequence of location points) with the query area should be evaluated. Even reducing the space size by building a hierarchical structure of trajectories such as prefix-trees, cannot effectively reduce the cost of computing range queries. A hierarchical structure, with a height h and branching factor a takes $O(a^h)$ cost to compute a range query. This problem underlines the importance of spatial histograms to efficiently compute the correct answer of this query type with the order of $O(k \ll N)$ where k is the size of range

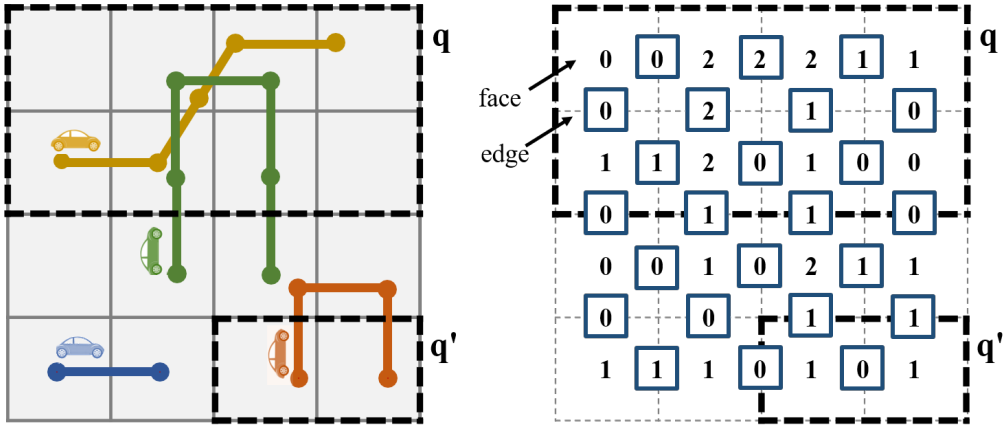


Figure 3.3: A trajectory dataset (left) and its corresponding spatial histogram (right). Circles on each trajectory represent location points. q and q' indicate two spatial range queries.

query. Without loss of generality, we focus on range queries with a rectangular target area — i.e., the axes are parallel and aligned with the boundary of cells in the grid domain. The results can be generalized to any convex query type. For queries that have partial overlap with a cell in the histogram, we consider the entire cell as part of the query area (i.e., expanding the query area). The range query is further discussed after introducing the spatial histogram data structure.

3.2.2 Spatial Histogram

Figure 3.3 (left) displays a dataset with four trajectories in a 4×4 grid domain, and Figure 3.3 (right) shows the corresponding spatial histogram of the dataset. A spatial histogram is a multi-dimensional data structure that efficiently summarizes the distribution of trajectories in a grid area [14]. Regardless of the movement direction, the histogram counts the numbers of trajectories overlapping grid cells. It has two components: *face* and *edge* matrices. The face maintains for each cell the count of overlapping trajectories, and the edge records the count of trajectories crossing each edge. Each trajectory contributes to the face and edge counts by increasing their total by one. If a trajectory crosses a vertex (i.e., intersection of edges) or travels along an edge, we consistently assign it to one of the corresponding faces. In Figure 3.3 (right), the face values are shown inside the cells, and the edge values are on the edges of cells (blue boxes). We formally define a spatial histogram as follows.

Definition 3.1. *(Spatial Histogram) Given a trajectory dataset $D \subseteq \mathscr{U}$ and a grid area with cardinality $r \times c$, a spatial histogram is represented by $H = \{F, E_v, E_h\}$ consisting of face F, ver-*

tical edge E_v and horizontal edge E_h matrix components, where $F \in \mathbb{N}_0^{r\times c}$, $E_v \in \mathbb{N}_0^{(r-1)\times c}$ and $E_h \in \mathbb{N}_0^{r\times(c-1)}$. By defining the domain $\mathscr{H} = \{\mathbb{N}_0^{r\times c}, \mathbb{N}_0^{(r-1)\times c}, \mathbb{N}_0^{r\times(c-1)}\}$, we can represent a spatial histogram as a mapping $H : D \to \mathscr{H}$ from trajectories to the counts.

From now on, for convenience, E is used as the edge instead of the two components E_v and E_h with appropriate operations, i.e., $E = \{E_v, E_h\}$. We use $|H|$ to show the spatial resolution of H. Let a dataset $D \subseteq \mathscr{U}$ contains n trajectories. Then the corresponding H supports n trajectories which can grow to the size of universe $\mathscr{U}$. A spatial histogram which is normalized to n, represents a distribution over the universe $\mathscr{U}$ of trajectories. In a normalized H, each face/edge value represents a proportion of trajectories intersected with the face/edge. The edge component maintains an aggregated information about the sequentiality of cells a trajectory has intersected. Each time a trajectory passes from cell a to cell b, the value of the corresponding edge between the cells are incremented by one, and the other edges remain intact. As it is later shown, this information is crucial for answering range queries. The edge component is the particular feature of spatial histograms that enables this structure to be efficient and correct in computing range queries.

Given a spatial histogram, the query q in Figure 3.3 counts the distinct trajectories T by subtracting the sum of edges inside the query area (q_E) from the sum of faces inside the area (q_F), i.e., $T = 9 - 7 = 2$. We formally define a spatial range query as follows:

Definition 3.2. *(Spatial Range Query) Given a spatial histogram $H = \{F, E\}$, a spatial range query $q = \{q_F, q_E\}$ with $q_F \subseteq F$ and $q_E \subseteq E$ is a function $q : \mathscr{H} \to \mathbb{N}_0$. It counts the number of distinct trajectories as*

$$q(H) = q_F(F) - q_E(E),$$

where $q_F(F) = q_F \diamond F$ and $q_E(E) = q_E \diamond E$, and "$\diamond$" is the dot product defined over matrices (i.e., sum of element-wise multiplication of the matrices).

Note that the dot product only applies to the *inner edges and faces* in the query support area and the border edges are not included. It is important to recognize that a trajectory may have multiple intersections with a query area. The orange trajectory and q' in Figure 3.3 show an instance of this behaviour. This problem can also occur when the query area is not convex. The works in [60] [15] used the spatial histogram structure in definition 3.1 and improved it for such movement types in trajectories or query shapes. Our work is based on the spatial histogram defined above, which can

D	Trajectory data set, $D \subseteq \mathscr{T}$ and $\|D\| = n$
H	Spatial histogram related to D, $H = \{F, E\}$, $H \subseteq \mathscr{H}$
E	The horizontal and vertical edges in H, $E = \{E_v, E_h\}$
j	Index of a cell in histogram, $j = (row, col)$
$H[j]$	A cell in H with a face $f \in F$ and four edges $e \in E$
f_j	The face component of cell $H[j]$
e_{ij}	The incident edge of two adjacent cells $H[i]$ and $H[j]$
H'	Synthetic spatial histogram
H''	Intermediate synthetic histogram in the synthesizing process
T	Total number of iteration in the synthesizing process
Q	Set of queries $q \in Q$ that $q = \{q_F, q_E\}$
$q[j]$	The jth element of query, $q[j] \in \{0,1\}$
$q(H)$	Computing the query q on histogram H

Table 3.1: Summary of notation.

be generalized to the advanced versions.

Having a trajectory dataset D mapped to a spatial histogram H, and a set of spatial range queries $Q = \{q_i | i \in \{1,2,...,m\}\}$ with size $|Q| = m$, we develop a mechanism to learn the distribution of H privately. The output of mechanism is an estimated spatial histogram H' while preserving privacy and providing accurate answers for range queries. The notations of this chapter are summarized in Table 3.1.

3.2.3 Differential Privacy Settings

According to differential privacy definition, if the difference between the probability of mechanism outputs, in presence/absence of a record in the input dataset, are bounded (controlled by ε), the mechanism is ε-differentially private. In a trajectory dataset D and in turn in a spatial histogram H, the neighbors are achieved by adding or removing a trajectory from the dataset. From now onward, we use the trajectory dataset D and the spatial histogram H interchangeably, since H represents a mapping of the trajectory dataset D and changing a record in D would directly affect the counts in H.

In range queries, the sensitivity of query q captures the maximum absolute difference of answers over any pair of neighbor datasets. Adding/ removing a record from a trajectory dataset/spatial histogram changes the answer at most by one. Thus, the sensitivity of the range query is 1. Laplace mechanism (LM) and Exponential mechanism (EM) are used in this chapter to

implement differential privacy. The Laplace mechanism adds a Laplace noise to the query answer with mean 0 and scale $\Delta q/\varepsilon$. Since we need to choose the best query to optimize the release data Exponential Mechanism is employed for selecting the next query. The sensitivity of utility function in the Exponential Mechanism, denoted by Δu, is defined to get $\Delta u \in [0,1]$ and bound the difference between probability of selecting q on any two neighboring datasets.

By bounding the sensitivity (Δq in Laplace mechanism or Δu in exponential mechanism), the privacy parameter, ε, plays a key role in the scale of injected noise.For a trajectory dataset D, if a differentially private mechanism adds noise to each point in a trajectory $t \in D$, the serial composition would happen. According to the serial composition property, the privacy budget ε is distributed between $|t| = l$ points in t that each point receives ε/l share of the budget. In the real dataset, the l might be large, which causes a considerable noise value to each point, that in turn may seriously disrupt the utility of trajectories. Another approach is considering a trajectory as a single point in a high dimensional space and then add noise to this point. This approach works for small trajectory datasets and increasing the size of dataset issues the same problem in the former approach [19].

By mapping a trajectory dataset into a spatial histogram H, each trajectory breaks into a set of face and edge counts. This key feature changes the problem into a parallel composition in which each count in the histogram can be treated separately in receiving the privacy budget. This simple mapping makes our private mechanism independent of the length of trajectories. However, a naive approach in adding noise may disrupt the sequential dependency in trajectories which degrades the utility of sanitized histogram.

To preserve the sequential dependency, our mechanism enforces a specific feature of spatial histogram data structure, which is related to edge and face counts. In a spatial histogram, *an edge value can be at most equal to the minimum of its two adjacent face counts.* For example, if two adjacent faces are f_e with count 4 and f'_e with count 8, the count of incident edge e between f_e and f'_e can be at most 4. Because the edge component maintains the information when trajectories cross the adjacent cells and its count does not increase if there are no corresponding face counts in the adjacent cells. We call this property a *dependency constraint*: given a histogram $H = \{F, E\}$, $\forall e \in E$ and its adjacent faces $f_e, f'_e \in F$, we have $c(e) \leq min\{c(f_e), c(f'_e)\}$ where $c(x)$ represents the count of x. We propose a private mechanism that guarantees to satisfy the dependency constraint

Algorithm 1 Private Spatial Histogram (PriSH)

Input: Histogram H with resolution $|H|$,
Query set Q,
Number of iterations I,
Privacy parameter ε **Output**: Synthetic histogram H'

1: Let H' be a spatial histogram with resolution $|H|$
2: Let H'_0 be initialized with a uniform distribution over $\mathscr{H}$
3: **for** i in I **do**
4: $wq_i = selectQuery(H, Q, \varepsilon/2I)$
5: $na_i = wq_i(H) + Lap(2I/\varepsilon)$
6: Save (wq_i, na_i) in $infList$
7: $H'_{i+1} = update(H'_i, infList)$
8: **end for**

return: H'

in the published histogram.

3.3 Private Mechanism

Algorithm 1 shows the Private Spatial Histogram PriSH mechanism. It receives a spatial histogram H of size d and a set of range queries Q. The histogram H is a mapping from a raw trajectory dataset D. The goal is to learn the distribution of H privately and publish an estimated histogram H' to accurately answer the queries Q. Each query $q \in Q$ provides information about the true distribution and leads the estimation direction in each iteration. The learning process starts with a uniform distribution of the histogram universe $\mathscr{H}$ assigning an equal weight for all values in H'. In each iteration $i \in I$, the *selectQuery* selects the most informative query $wq_i \in Q$ using exponential mechanism . The noisy answer of wq_i given by Laplace mechanism , na_i, is then used to update the estimation, in the function called $update$.

The critical point about PriSH is the type of output. We do not add noise to the trajectories in D or the query set directly. Instead, we estimate the distribution of trajectories in the form of a histogram using our ε-differentially private mechanism so that the estimation has high utility in answering each query $q \in Q$. Converting trajectories into a histogram makes PriSH independent of trajectory lengths, and then the utility of output is not highly data-dependent.

The remaining of this section presents the technical details of the query selection strategy ($selectQuery$) and the updating strategy ($update$) in our mechanism. We define a utility func-

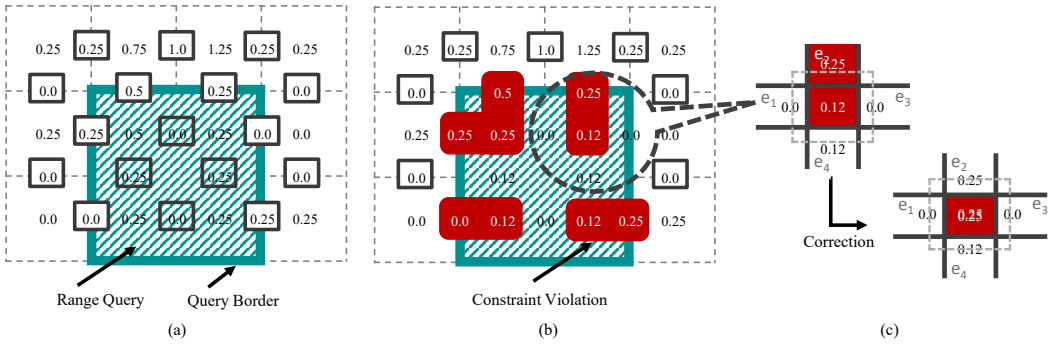


Figure 3.4: Dependency constraint violation by $MWUR$: (a) the estimated H'_{i-1} in iteration i and the wq_i which is shown in green, (b) how the rescaling causes violations in new estimation H_i, and (c) our correction strategy for the violations.

tion for spatial range queries based on the query errors. Using the selected query, we introduce a constraint-aware update rule that improves the estimation and incrementally satisfies the sequential dependency in the histogram. Following sections analyze the complexity, formal guarantees and performance of our mechanism.

3.3.1 Query Selection Strategy

Without any limitation, the algorithm could use as many queries as it needs to learn the distribution. However, privacy concerns restrict the algorithm to have a small number of iterations. In each iteration, the mechanism have to select a noisy optimal query that provides the most improvement for the estimation. The optimality is measured by the absolute query error on the estimation H'. Intuitively, a query with larger error provides a better chance to improve the estimation, and in practice, larger queries often have larger errors. The error of $q \in Q$ defines as L_1 norm of the difference in query answer on H' and H, i.e.,

$$qerror = ||q(H) - q(H')||_1.$$

To select the optimal query, the Laplace mechanism is not practical since it divides the privacy budget ε between queries and disrupts the quality of query selection in large Q which may affect the accuracy of estimation. We use the exponential mechanism to privately select a query as it saves the ε by applying the parallel composition property. The critical point, however, is defining a utility function to represent the selection criteria correctly. We consider the query error $qerror$ as

Algorithm 2 Update Function

Input: Estimation H'_{i-1},
$inflist$, list of measures to improve the estimation **Output**: Updated histogram H'_i

n = number of trajectories in H'_i
for (wq, na) in $inflist$ **do**
$qerror = na - wq(H'_{i-1})$
$H'_i[x]_{\forall x \in |H'|} = H'_i[x] \times e^{wq[x] \times qerror/2n}$
if $qerror < 0$ and $isVio() == True$ **then**
$corrVio(wq, H'_i)$
end if Normalize H'_i to n trajectories
end for
return: H'_i

the utility of each query. In each round $i \in I$, the $selectQuery$ computes the utility of each $q \in Q$ and let the exponential mechanism to select one privately. After selecting the most informative query wq_i, the algorithm uses its noisy answer to update the estimation. The Laplace mechanism with parameter $2I/\varepsilon$ is used to perturb the real answer of wq_i.

3.3.2 Updating the Estimation

The $update$ function in Algorithm 2 is the heart of PriSH as it aims to correctly capture the sequential dependency among the cells while updating the values. The goal of the update function is to improve the estimation H' toward the true distribution of H. The selected query wq_i and its noisy answer na_i in iteration $i \in I$, provide measures for the update. The query itself specifies the area in histogram H'_i that we have new information to update, and its noisy answer provides a measure to compute the error of query on H'_i and then identify the updated scale.

Since we are estimating the counts, the update technique has to ensure the non-negativity of updated values. Techniques like subtraction or addition may cause negative values that are not meaningful for counts. Hardt et. al. [74] introduced the *multiplicative weights update rule* ($MWUR$) for 1D-histograms. By starting from a uniform weight for values, the technique updates the weights by multiplying them with a scaling factor sf_i in iteration $i \in I$. Using the multiplication and an exponential function to determine the sf_i, $MWUR$ ensures the non-negativity of updated values. We extend the $MWUR$ to 2D-histograms such as spatial histograms in which the histogram maintains the counts as well as their locations in the spatial space. The scaling factor sf_i for each

value $x \in H'$ is proportional to the query error, i.e., $qerror_i = na_i - wq_i(H'_{i-1})$:

$$sf_i = e^{wq_i[x].qerror_i}.$$

The $wq_i[x]$ is the weight for x^{th} element of H' in wq_i. The values *inside* the query box are scaled by sf_i and others remain intact. After scaling, the $MWUR$ re-normalizes the weights in H'_i to the number of trajectories, n, to re-adjust the importance of the estimated values in shaping the true distribution. For a query with positive (negative) error, the technique scales up (down) the estimation to reduce the error. Hardt et. al. [74] showed that the update rule improves the utility with a factor of $qerror_i^2$ and the algorithm can converge to an accurate estimation in a linear number of iterations $I \ll n$, where n is the total number of trajectories aggregated in H.

Although $MWUR$ generates non-negative values, it is not directly applicable to spatial histograms. Recall the dependency constraints (section 3.2) that represent the sequential dependency in trajectories. $MWUR$ may violate the constraints when wq_i does not cover the entire histogram. Figure 3.4(a) specifies a range query area and its borders. In Figure 3.4(b), given a scale factor $sf_i = 0.5$, the face and edge values inside the query box are scaled-down. Scaling has made some face values less than the adjacent edges (red areas) because $MWUR$ is not data-aware and does not consider sequential dependencies among cells when updating the values.

The constraint violation occurs when the $qerror_i$ is negative. A negative $qerror_i$ shows that the estimation is larger than the correct values and needs to be reduced. In such cases, the $MWUR$ generates a $sf \in (0,1)$ which leads to downscaling and often making faces smaller than edges in the query borders, i.e., violating the dependency constraint. While scaling up, the face values increase without an effect on the dependency constraints on the borders.

We introduce an *enhanced MWUR*, $eMWUR$, in which the update preserves the dependency constraints. In case that $qerror_i$ implies down scaling, the $eMWUR$ calls *isVio* function shown in Algorithm 3 to check whether any (edge, face) pairs in wq_i borders violate the constraints (see Figure 3.4(a) and Figure 3.4(b)). Having E'_b as border edges of $wq_i = \{q_{F'}, q_{E'}\}$, the *isVio* returns *True* if the size of

$$v = \{(e, f_e) \mid c(e) > c(f_e), \forall e \in E'_b, f_e \in q_{F'}\}$$

is not zero, where f_e is the adjacent face of edge e. Otherwise, the *isVio* returns *False*. Note that

Algorithm 3 isVio Function

Input: $H'_i = \{F', E'\}$, $wq_i = \{q_{F'}, q_{E'}\}$, the selected query area **Output**: Consistency violation status

```
Let E'_b ⊆ E' be the border edges of wq_i
for e ∈ E'_b and f_e ∈ q_F' do
    if c(e) > c(f_e) then                    # c(x) represents count of x
        return: True
    end if
end for
return: False
```

E'_b, $q_{E'}$, and $q_{F'}$ are subsets of the estimated histogram $H' = \{F', E'\}$ components.

If a violation happens, the *corrVio* function corrects pair values $(e, f_e) \in v$ to satisfy the dependency constraints. *corrVio* follows the requirement in dependency constraint and heuristically *increases the violated face value to the maximum of edge values corresponding to the face* — i.e., $c(f_e) = max\{c(e_1), c(e_2), c(e_3), c(e_4)\}$ in Figure 3.4(c). Algorithm 4 shows this effective strategy that efficiently corrects the violations without additional computational cost.

Note that the other conceivable approach for correcting violations could be finding the closest histogram to the updated H' that satisfies the dependency constraints. Since the mechanism updates a part of estimation in each iteration, the correction should be applied in each iteration. Otherwise, the violation accumulated during the iterations takes the estimation far from the actual distribution, which results in the low utility of output. If an optimization approach is applied for correcting the constraint violation, it would impose extra computational cost per iteration and degrades the performance of the mechanism. In contrast, our heuristic strategy ensures satisfying the constraint without extra computational cost (see Section 3.4).

PriSH saves the (wq_i, na_i) pairs in a list called $inflist$ to incrementally improve the utility of estimation using the selected queries. As shown in Algorithm 2, after each update, H'_i is normalized to re-adjust the weight of all elements. Given the $inflist$, the update phase can be continued (let k times) without any privacy cost until it ensures H works accurate on queries in $inflist$. We have used this advanced setting in our experiments.

Algorithm 4 corrVio Function

Input: $H'_i = \{F', E'\}$,
$wq_i = \{q_{F'}, q_{E'}\}$, the selected query area
Output: Consistent histogram H'_i

```
Let E'_b ⊆ E' be the border edges of wq_i
for e ∈ E'_b and f_e ∈ q_F' do
  if c(e) > c(f_e) then                    # c(x) represents count of x
    c(f_e) = max of f_e corresponding edges
  end if
end for
```

3.4 Running Time

Input to Algorithm 1 is a spatial histogram and it generates a differentially private version of input histogram preserving statistical properties of the input histogram. The running time of PriSH is $O(|H||Q| + I.k|H||Q|)$, i.e., linear to the resolution of input spatial histogram, $|H|$, and the learning iterations $I.k$. The computational cost of PriSH depends on three steps: query selection in $selectQuery$, perturbing the answer using Laplace mechanism , and updating the estimation in $update$.

Give the query set Q, the $selectQuery$ computes the $qerror$ for all $q \in Q$ which costs $O(|H|)$ per query and then, $O(|H||Q|)$ as total. The noisy answer of wq can be achieved by a cost of $O(|H|)$ which is nearly a large constant value. Updating the contributed area in wq takes $O(|H|)$ in $update$. The mechanism continues the learning process for I iterations and progressively makes the $inflist$ by selected queries. In worst case, $|inflist| = |Q|$. Since we use the advanced version, inside each iteration $i \in I$, the algorithm repeats the update function for k times to optimize the estimation for all $(wq, na) \in infList$. Let the wq target $r \times c$ cells in the histogram. Since the $isVio$ and $corrVio$ evaluate the borders of query box, each function just takes $2 \times r \times c$ which is a small constant. Therefore, the update function costs $O(I.k.|H|.|infList|) \leq O(I.k.|H|.|Q|)$ for each iteration.

The number of iterations I, in worst case, can be as large as the size of input query set Q and should be considered in computing time complexity. However, it is observed in the experiments that Prish can achieve high accuracy output with a much smaller number of iterations than the size of the query set. In practice, the size of $inflist$ (after I iterations) is much less than the query set, $I = |infList| \ll |Q|$ and k is a small constants. In total the three steps have $O(|H||Q| + k|H||Q|)$ computational cost in each round $i \in I$. Since the number of iterations, I, is a constant value

and much smaller than the size of the query set Q, its effect on the complexity is negligible. We highlight that for PriSH most of the computations could be done in parallel. Selecting the query in $selectQuery$ and updating elements in $update$ can easily be computed in parallel, which significantly improves the running time. As the running time shows: PriSH is independent of the length of trajectories and the size of input trajectory dataset.

3.5 Formal Guarantees

Since PriSH is constructed based on MWEM, we evaluate its privacy and accuracy guarantees comparing to the base algorithm. We analyze the effect of our enhanced update rule $eMWUR$ and show that PriSH provides the same guarantees as MWEM.

Theorem 3.1. *PriSH satisfies ε-differential privacy.*

Proof. The exponential mechanism in $selectQuery$ costs $\varepsilon/2I$ and the Laplace mechanism scales the noise by $2I/\varepsilon$. The $eMWUR$ improves the estimation and corrects the violations independent of the input dataset. Due to the sequential composition, the privacy cost would be ε across I rounds. □

The maximum error of estimation across all $q \in Q$ determines the accuracy bound of PriSH. The accuracy of PriSH is affected by two factors: (1) the *perturbation noise* to satisfy differential privacy requirements, (2) the *estimation noise* imposed by the quality of $eMWUR$ to improve the estimation. We show that our enhanced update rule achieves the same quality as $MWUR$ and the accuracy bound of PriSH is as follows.

Theorem 3.2. *Given a spatial histogram H, for any $Q, I \in \mathbb{N}$, $\varepsilon > 0$, with probability $\geq 1 - 2I/|Q|$, PriSH estimates H' so that*

$$\max_{q \in Q} |q(H) - q(H')| \leq 2n\sqrt{\frac{log|H|}{I}} + \frac{10Ilog|Q|}{\varepsilon}$$

Proof. We show that our $eMWUR$ does not affect the accuracy bound of MWEM and hence, PriSH has the same bound. The appendix A in [46] explains the proof of accuracy for MWEM. Without further amendments, the proof carries over from 1D to 2D for the spatial histogram.

The proof first shows that the perturbation error imposed by the exponential mechanism and the Laplace mechanism is small. second, the worst-case approximation error of $MWUR$ is analyzed and then the accuracy bound is computed with respect to the two error. To capture the estimation improvement, in second step, the relative entropy is used which shows the difference between the H and H' after each update round:

$$\psi_{i-1} - \psi_i \geq \left(\frac{q_i(H'_{i-1}) - q_i(H)}{2n} \right)^2 - \left(\frac{na_i - q_i(H)}{2n} \right)^2, \tag{3.1}$$

where na_i is the noisy answer of $q_i(H)$. The effect of our correction strategy in PriSH appears in this step. We the $eMWUR$ does not change the bound of estimation error.

After scaling each $x \in H$ by sf_i, $eMWUR$ multiplies the x by a correction multiplier α_x where $\alpha_x \geq 1$. Considering a maximum multiplier in each round, we define α as the maximum bound on the correction multipliers across all rounds, i.e.,

$$\alpha = \max_{i \in I} \max_{x \in H'} \alpha_x.$$

According to the definition of relative entropy, for $i \in I$, we have:

$$\psi_{i-1} - \psi_i = \sum_{x \in H} H[x] log \left(\frac{H'_i[x]}{H'_{i-1}[x]} \right) / n. \tag{3.2}$$

The ratio $\frac{H'_i[x]}{H'_{i-1}[x]}$ can be written as $\frac{\alpha}{\beta_i} sf_i$ where $sf_i = exp(q_i[x]\eta_i)$ with $\eta_i = (na_i - q_i(H'_{i-1})/2n$, and β_i is the re-normalization factor. Expanding the equation gives

$$\psi_{i-1} - \psi_i = log(\alpha) + \frac{\eta_i}{n} q(H) - log(\beta_i). \tag{3.3}$$

Using the Taylor expansion of sf_i and that $|q_i(x)\eta_i| \leq 1$ and $q_i(x)^2$, the β_i bounds is:

$$\begin{aligned}
\beta_i &= \sum_{x \in H} (\alpha . sf_i . H'_{i-1}[x])/n && (3.4)\\
&\leq \sum_{x \in H} \frac{\alpha}{n}(1 + q_i(x)\eta_i + \eta_i^2) H'_{i-1}[x] && (3.5)\\
&= \alpha(1 + \frac{\eta_i}{n} q_i(H'_{i-1}) + \eta_i^2). && (3.6)
\end{aligned}$$

Inserting this bound to the equality (2) and that $log(1+x) \leq x$ gives:

$$\begin{aligned}
\psi_{i-1} - \psi_i &\geq \frac{\eta_i}{n} q(H) - log(1 + \frac{\eta_i}{n} q_i(H'_{i-1}) + \eta_i^2) && (3.7)\\
&\geq \frac{\eta_i}{n} q(H) - \frac{\eta_i}{n} q_i(H'_{i-1}) + \eta_i^2 && (3.8)\\
&\geq \eta_i (q_i(H) - q_i(H'_{i-1})/n - \eta_i^2. && (3.9)
\end{aligned}$$

By introducing the definition of η_i it gives the inequality (1) which is the same as MWEM bound. The remaining of the proof follows the appendix A in [46]. □

Considering the range of query answer $q(H)$ which is $[0,n]$, we can bring the error smaller than n by choosing $I \geq 4log|H|$. Note that this is the worst-case accuracy bound. As the experiments show, PriSH works well in different settings and our correction strategy improves the accuracy of the estimated data in real settings.

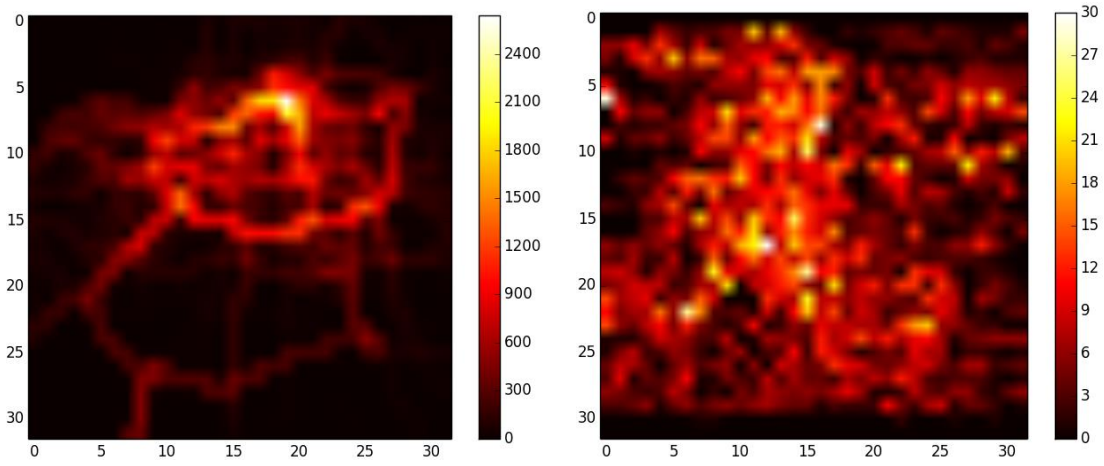


Figure 3.5: The heatmap of a uniform sample from PORTO-TAXI (left) and MELB-CAR (right) datasets.

3.6 Experiments

We evaluate our mechanism based on (1) its accuracy in answering spatial range queries in different settings, and (2) the utility of estimated distribution compared to the actual distribution.

Datasets. Two trajectory datasets are used in the experiments. PORTO-TAXI is a real dataset describing trajectories of 442 taxis operating in the city of Porto, Portugal [73]. The dataset contains more than 1.7 million trajectories for a year. We chose this dataset as it features a large number of trajectories with various lengths in a large metropolitan area. MELB-CAR is a synthetic, uniformly distributed dataset that is generated by the Minnesota Web-based Traffic Generator (MNTG) [75]. We simulated 1,000,000 vehicle trajectories for 20 time units in the city of Melbourne, Australia. To create a spatial histogram for each of these datasets, we considered a square of 100km^2. Figure 3.5 shows the distribution of (a uniform sample from) trajectories in PORTO-TAXI (left) and MELB-CAR (right). In PORTO-TAXI the trajectories are mostly concentrated in the city center, and the traffic becomes sparser as it is further away from the center. In MELB-CAR, however, there is no traffic center and trajectories are more uniformly distributed. Having a single center with a few trajectories in the outskirts of PORT-TAXI, makes it difficult for an algorithm to learn the distribution of data accurately. We evaluate the accuracy of PriSH using samples from this dataset. For the utility analysis, we evaluate the effect of distribution by

comparing the output for the two datasets.

Queries. We run experiments using queries that are chosen randomly. For each query, a rectangle with $x = [x_1 - x_2]$ and $y = [y_1 - y_2]$ exes is generated where the intervals are sampled uniformly at random. Other than mentioned explicitly, we use a query set Q of size 16,000 in the experiments.

Baselines. The state-of-the-art mechanisms in decomposing the space [51] [46] are used as baselines. DAWA mechanism [51] considers the query properties as well as data properties in privatizing the decomposed space. Using a density-based partitioning, DAWA first decomposes the space into cells, and then adds a correlated noise to the cell values with respect to the correlation between queries in the given query set. The aim in DAWA is optimizing the accuracy of published data in answering a given query set. MWEM [46] is a query aware that optimizes the output for a class of query, not limited to a given query set. By laying a grid with a fixed cell size on the spatial space, MWEM iteratively learns the density of cells using the noisy answer of queries which are randomly selected from the class of query. Both DAWA and MWEM approaches adaptively partition the space to optimize the error of published data in answering queries. However, the proposed mechanisms are designed for location point datasets rather than trajectories.

Measures. We use the average L_1 error per query as a metric for accuracy and the *relative entropy* for the utility of generated distribution. The latter measure quantifies the divergence between the true distribution H/n versus the estimated one H'/n:

$$RE(H||H') = \sum_{j \in |H|} H[j] log(\frac{H[j]}{H'[j]})/n,$$

where $|H|$ is the resolution of H and n is the number of trajectories in the given dataset. The relative entropy is also known as Kullback-Leibler Divergence (KLD) and is used in MWEM results in [46] to measure the utility.

Default settings. The grid resolution in PriSH and MWEM is assumed to be a power of 2. Depending on the application, the histogram may have a different level of resolution. Suppose a dataset contains taxi trajectories in a city center. Considering a speed limit of $50km/h$ for the city, a grid with the resolution level 8 would aggregate the trajectory points in $2^8 \times 2^8$ cells with area of $39m^2$ for each cell, or the level 9 gains $2^9 \times 2^9$ histogram with about $19.5m^2$ cell area. In our experiments, other than mentioned explicitly, the resolution level is set to 8 regarding the

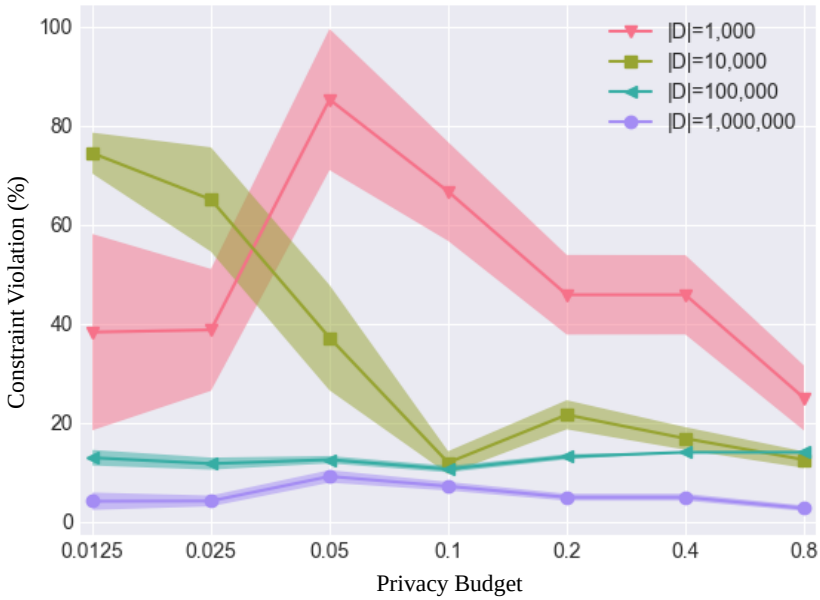


Figure 3.6: Dependency constraint violation in the published data estimated by *MWUR* strategy.

speed limit of cars in the two cities which the datasets are taken from. The number of iterations are chosen to be $I = \{10, 12, 14, 16, 18, 20\}$ for PriSH and MWEM, and the best result is reported. The reported result is an average of 5 independent repetitions of the algorithm. For the learning parameter k in the advanced update function (explained in Section 3.3), we found $k = 10$ works well for PriSH. According to [51], the ratio of dividing the privacy budget ε, between the steps of DAWA is set to $r = 0.25$. For MWEM and hence PriSH, we follow the setting in [46] and consider $r = 0.5$ to assign equal ε for selecting query and then learning the distribution.

3.6.1 Accuracy Evaluations

Constraints and correction strategy. Figure 3.6 shows the importance of considering dependency constraints in estimating the distribution. The figure depicts the percentage of histogram components (edge and faces), in the estimated distribution by *MWUR* update rule, which violates the dependency constraint. From sparse to dense datasets, scaling from 1000 trajectories to 1,000,000, learning the distribution of a spatial histogram without considering the constraint, breaks the sequential dependency between the histogram components. However, increasing the size of dataset leads to higher values in the histogram, and in turn, reduces the sensitivity of values

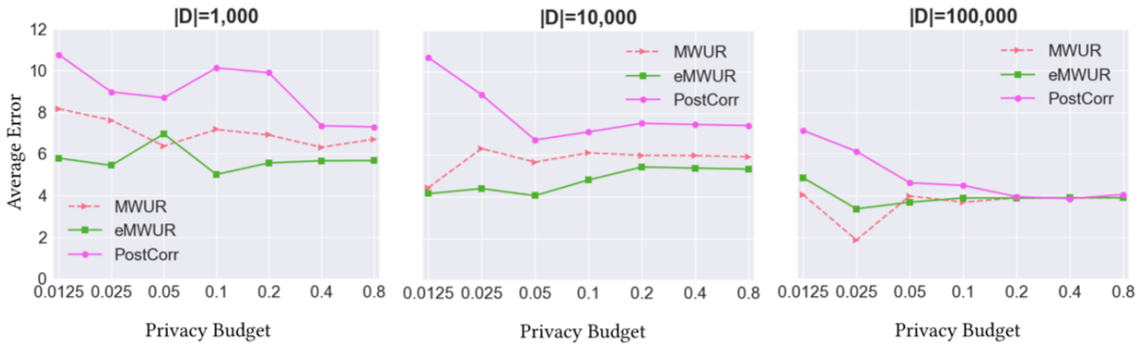
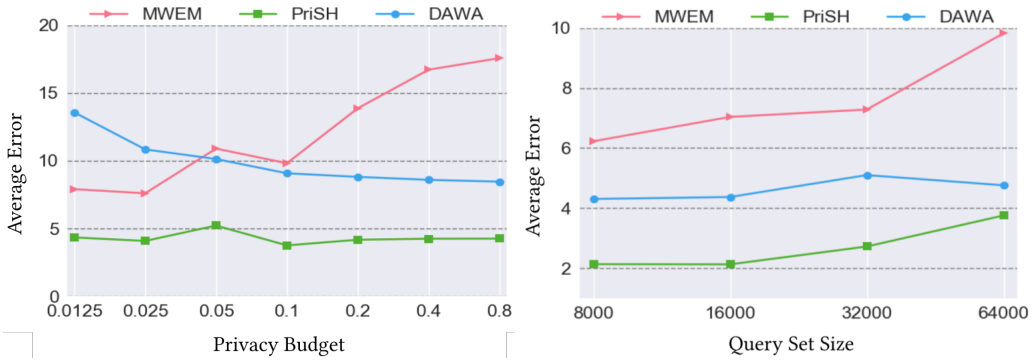


Figure 3.7: Accuracy of estimation by *MWUR*, *eMWUR*, and *PostCorr* strategies for correcting constraint violations on datasets of size 1000, 10,000 and 100,000 trajectories. The dashed graph indicates the result is not guaranteed to satisfy the dependency constraints, whereas the solid lines guarantee the spatial constraints.



(a) Real dataset D (b) Synthetic dataset D'

Figure 3.8: Contrasting the accuracy of PriSH, MWEM and DAWA.

to the noise. This can be seen in the percentage of violations in larger datasets. It is important to note that the estimated histogram distribution must have no violation. Otherwise, the histogram does not comply with the definition of the spatial histogram.

Figure 3.7(a)-3.7(c) compare three correction strategies: *MWUR* (used in MWEM), *eMWUR* (introduced in PriSH), and *PostCorr* strategies in datasets of size 1000, 10,000, and 100,000 respectively. The *PostCorr* strategy, in contrast to *eMWUR*, does not correct the violation in each iteration $i \in I$, and instead rectifies the violations in the final estimation generated in round I.

We see that *eMWUR* outperforms the *PostCorr* strategy, especially in small datasets. The accumulation error caused by constraint violations during the learning process takes the estimation far from the actual distribution. Therefore, pushing the estimation into the feasible space in *PostCorr* strategy leads to the worst estimation. *eMWUR* also improves the accuracy of *MWUR*,

especially in smaller εs and datasets with sparse trajectories. Since *eMWUR* is heuristic, in some settings, it may slightly degrade the accuracy, but in return, we obtain a *valid* estimation (satisfying dependency constraints).

Baselines. The closest algorithms to PriSH are MWEM [46] and DAWA [51] (see Section 3.2). For MWEM and DAWA, the histogram and queries are converted to 1D using Hilbert curves. Since these algorithms do not consider the dependency constraints, the query error in MWEM and DAWA is computed using just the face values. Figure 3.8 contrasts the accuracy of algorithms in changing the privacy budget (with a fixed query set $|Q|$ =16,000) as well as the size of query set (with a fixed privacy budget $\varepsilon = 0.1$).

As the results show, our mechanism PriSH significantly outperforms MWEM and DAWA in both settings. The accuracy of *PriSH* is almost stable for different εs. In contrast, MWEM is sensitive to the ε and its accuracy is affected by that as it does not consider the dependency among the cells in the histogram. The higher accuracy of MWEM for lower εs is due to the cancellation of noise. On the other hand, the accuracy of DAWA improves in higher εs because DAWA optimizes the output with respect to the given Q. However, DAWA cannot ensure high accuracy in answering a new set of queries Q', while MWEM and PriSH have high accuracy for arbitrary range queries. In addition, there is no guarantee that the data published by DAWA satisfies the dependency constraints. DAWA is almost stable with respect to the size of Q due to the optimization in this mechanism. The accuracy of PriSH slightly decreases in large Qs, which shows the effect of noisy query selection in the learning process. However, even in large query sets, PriSH is able to generate a valid histogram which answers range queries significantly more accurate.

Figure 3.9 compares the accuracy of published histogram on a random Q' of size 1000. We use a sparse sample $|D|$ =1000 from PORTO-TAXI and the accuracy of mechanism are evaluated in different resolution levels $L = \{3,5,7,9\}$ of the spatial histogram H. As the results show, PriSH accurately answers the random queries in all resolution levels. In contrast, DAWA does not have high accuracy on new queries, and its accuracy degrades considerably in higher resolutions. The reason lies first, in the construction of DAWA that optimizes the output for the particular query set given to the algorithm, and second, in the effect of edge values in a spatial histogram that DAWA does not consider them in the computations.

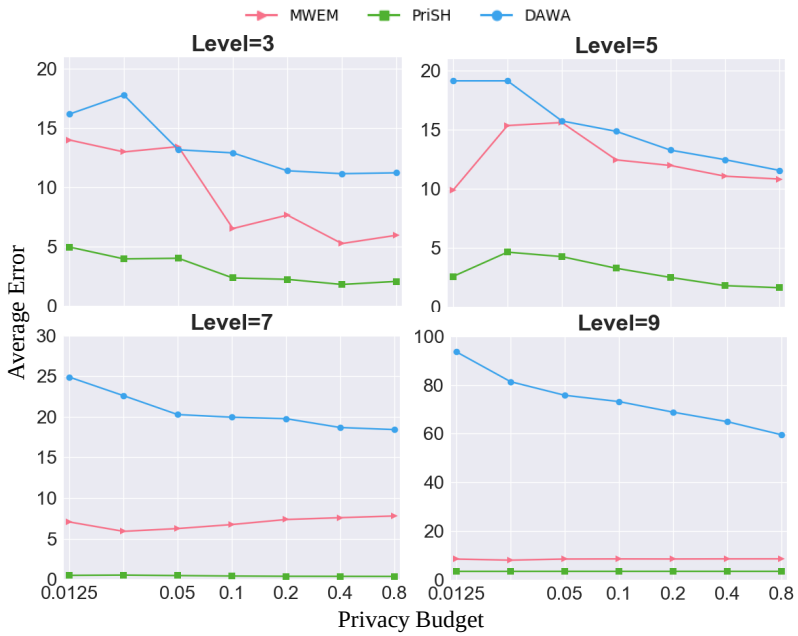


Figure 3.9: Contrasting the accuracy of PriSH and its competing algorithms on a new random query set $|Q'| = 2000$.

3.6.2 Utility Evaluation

Fig 3.10 shows the effect of distribution on the utility of the estimated distribution (measured by KLD) in PriSH and MWEM mechanisms. The PORTO-TAXI dataset has a skewed distribution in which the trajectories are mostly congested in a specific area, and most of the counts in the input histogram are zero. Such sparse datasets, especially when the size of the dataset is small, do not provide much information for the algorithm which makes the estimation more difficult. In contrast, in a uniformly distributed dataset such as MELB-CAR, each cell of histogram provides some information, and the algorithm has more information to conduct the estimation toward the true distribution. Increasing the size of the dataset, however, improves the information gain since the larger face/edge values are less sensitive to the noise, which in turn enhances the utility of generated distribution exponentially. We clearly observe this behaviour, specifically on a skewed distribution such as PORTO-TAXI. A similar scenario occurs in MWEM output. However, increasing the size to larger than 10,000 records cannot help to improve the utility. In addition, we do not observe better utility by increasing the ε in larger datasets, because MWEM cannot improve the estimation with wrong information from the true distribution. Note that the utility of MWEM is computed just using the face values, and no penalty has been applied for ignoring the edge values,

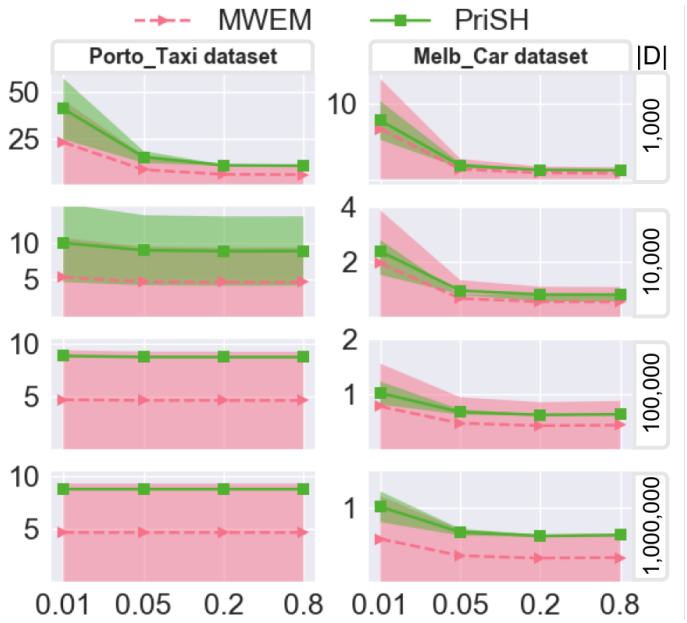


Figure 3.10: The utility of published data by PriSH vs. MWEM on samples of PORTO-TAXI and MELB-CAR datasets. The graphs show the mean value (solid and dashed lines) and the shaded area is the standard deviation. The dashed graph shows the output is not guaranteed to satisfy the dependency constraints whereas the solid line guarantees the spatial constraints.

which could seriously affect its utility on both datasets. On the other hand, we see that the utility of PriSH is slightly lower. The correction of constraint violations perturbs the learning process in PriSH, but it is essential due to spatial histogram requirements. However, we showed that PriSH has high accuracy in answering range queries, which is the main goal of publishing the spatial histogram.

3.7 Conclusion

We introduce PriSH, an ε-differentially private mechanism for synthesizing spatial histograms and privately answering range queries on trajectories. PriSH is based on MWEM [46]. We introduce a dependency constraint as a measure of sequential dependency in trajectories. Using this constraint, we introduce a learning process and update rule in PriSH, which accurately estimates the distribution of spatial histograms while ensuring the sequentiality property across the cells, and thus the consistency of the published histogram.

PriSH improves the sensitivity problem in privately answering queries on trajectory datasets.

Given a resolution level for a spatial histogram, PriSH can synthesize aggregated trajectories without assumptions on the trajectory movements, the lengths, or the size of the trajectory universe in the learning process [18, 19]. However, PriSH performance is sensitive to the correction strategy used for correcting the dependency constraint violations. Applying a heuristic strategy for stringent privacy levels may degrade the performance of PriSH. We view this as a future direction to introduce a correction approach to be tuned to the update scale function. We also aim to use the dependency between histogram components to rebuild a synthetic trajectory dataset using the published spatial histogram.

Chapter 4

A Differentially Private Algorithm for Range Queries on Trajectories

In Chapter 3, we introduced the first differentially private mechanism, named PriSH, for publishing spatial histograms. PriSH, takes a spatial histogram and a query set as input and utilizes the correlation between the queries to estimate the distribution of the original histogram privately. To maintain the consistency in the histogram, PriSH uses a heuristic *approach that may result in a histogram far from the original spatial histogram. The reason lies in the* heuristic *approach that locally ensures consistency and may lead to overcorrection. In this chapter, we propose a* data- and query-aware *mechanism that utilizes the trajectories density in different regions as well as the given queries correlation to estimate the optimal spatial histogram with significantly higher utility. Intuitively, trajectories are generally unevenly distributed across a city and adding noise uniformly will generally lead to a poor utility. our algorithm adaptively adds noise to the input data according to the given query set. It first privately partitions the data space into uniform regions and computes the traffic density of each region. The regions and their densities, in addition to the given query set, are then used to estimate the distribution of trajectories over the queried space, which ensures high accuracy for the given query set. Our query-aware strategy employs a linear programming approach to provide a guarantee on optimally consistent histogram which leads to significant improvement in the utility of results. We show the accuracy and efficiency of our algorithm using extensive empirical evaluations on real and synthetic data sets.*

4.1 Introduction

THE popularity of sensor-enabled devices (e.g., wearables and smartphones) has significantly advanced the capability of businesses to collect and analyze people's trajectories. Studying large-scale data sets and analyzing the movement patterns of individuals provide crucial

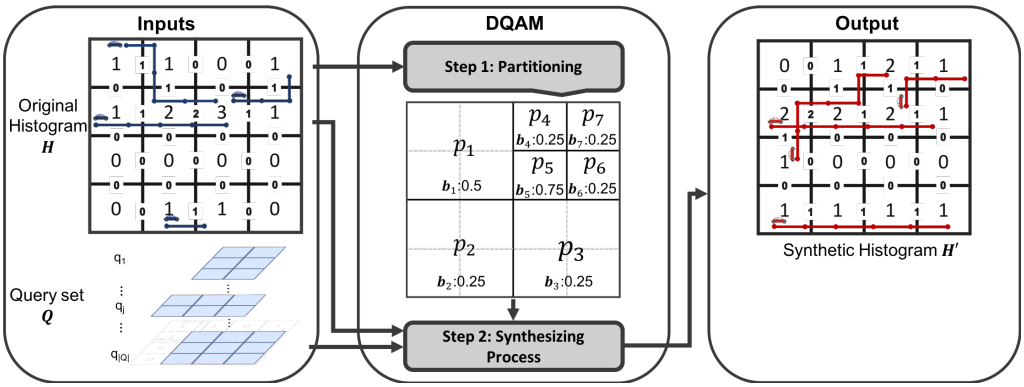


Figure 4.1: Overview of DQAM mechanism with an example. p_i and b_i depict the partition and its density, respectively.

insight for many applications (e.g., traffic management, route planning, urban planning, crime detection). Such applications critically rely on estimating the number of trajectories in an area. For example, in urban planning, computing the number of pedestrians and the flow of their movements provide significant information for the placement of public spaces (e.g., local parks, street spaces, plazas), pedestrian and bicycle paths, and public transport stations. A fundamental query type in studying the movement patterns is *range query* on trajectories (see Chapter 3) which counts the number of *distinct* trajectories intersecting the query area (2D space). However, computing the cost of this query type on raw trajectories is linear in the number of trajectories and the average length of the trajectories. To avoid such high computational cost *spatial histogram* (see Chapter 3) is commonly used to compute range queries efficiently. The cost of range queries using spatial histogram is linear in the size of the query area. A spatial histogram considers the data space as a grid with fixed cell size and aggregates the count of trajectories overlapping a cell or an incident edge of two cells in the grid. In other words, a spatial histogram maps the trajectory data into a set of counts representing the distribution of trajectories throughout the space.

Even though the value of spatial histograms in studying movement patterns is realized by companies, releasing such data to third parties is still a significant issue due to privacy concerns. Studies show that trajectories exhibit a high risk of being identified that aggregation or coarsening techniques as done in a spatial histogram, barely reduce their uniqueness [5], [57]. The key reason is that trajectories are time-stamped *sequences* of locations with a start and end points. For example, the start of a trajectory is often a home location. Thus, publishing a spatial histogram raises privacy concerns due to the uniqueness of peoples trajectories, particularly in sparse regions with

few trajectory counts.

A principled approach is needed to generate a synthetic spatial histogram with strong privacy guarantees of differential privacy [7] while ensuring high utility in practice. The utility refers to the accuracy in answering range queries. The approach can make use of a set of range queries to learn the distribution of data (called *query-awareness*). However, it should ensure the synthesized spatial histogram maintains high accuracy in answering any range queries on trajectories. A naive approach adds an equal noise to the cell counts in the histogram to ensure privacy. Since a trajectory may intersect multiple cells, the cells are *sequentially dependent*. Thus, naively scaling noise to satisfy differential privacy can violate the sequentiality between the cells, which is crucial in representing trajectories. Furthermore, the sensitivity of cell counts to the added noise varies depending on the *density* of trajectories in different regions of data space. A large region of space might be sparse with small cell counts which makes the cell counts in the region highly sensitive to the added noise. In contrast, a region might be small but very dense such that cell counts are large and highly noise resistance. The proposed approach should be able to ensure maintaining the data properties (called *data-awareness*) in the synthesized data.

In this chapter, we present *Data- and Query-Aware Mechanism (DQAM)*, a mechanism that synthesizes spatial histograms. As Chapter 3, DQAM takes a spatial histogram and a set of range queries as input and uses the correlation between the queries (i.e., is *query-aware*) to synthesize the histogram. In contrast to Chapter 3, DQAM identifies the density of different regions in data and make use of the density in scaling the added noise to the generated synthetic histogram (i.e., is *query-aware*). Also, DQAM generates the optimal synthetic histogram in each iteration of the synthesizing process that satisfies the sequentiality of cells. The proposed mechanism in Chapter 3 applies a greedy strategy to satisfy the sequentiality requirement among histogram cells, which may lead to overcorrection. While the work in Chapter 3 successfully deals with trajectories uniformly distributed in the data space, it can result in considerable loss of utility for trajectory data sets with a non-uniform distribution in different regions. DQAM is designed to deal with such realistic trajectory distributions by capturing the density of trajectories in different regions and using the density as a weighting measure: each region is assigned a weight relative to its density. To guarantee differential privacy, DQAM uses the well established Laplace mechanism [7] and Exponential mechanism [17] for adding noise in all parts of its computations.

Contributions. First, we propose a differentially private *Data- and Query-Aware Mechanism* (DQAM), a mechanism that publishes a spatial histogram to answer range queries efficiently. To the best of our knowledge, DQAM is the first data- and query-aware mechanism for range queries on sequential spatial data that reduces the error by a factor up to 7.4 compared to the current approaches on trajectory data sets. Second, we design an efficient data-aware algorithm with $nlog(n)$ time complexity in the number of cells in the spatial histogram that partitions the histogram into uniform regions. Third, we present a query-aware and differentially private strategy that captures trajectory sequentiality and synthesizes accurate output using the given queries and partitions.

Mechanism Overview

Differential privacy assumes a privacy budget for an algorithm that is described by a parameter ε. DQAM is a two-stage ε-differentially private mechanism for answering range queries on trajectories (see Fig. 4.1). The mechanism takes as input a set of range queries Q, and a spatial histogram H. The histogram H is a representation of trajectory counts throughout the data space. The query set Q is randomly generated and is large to cover the entire space of H. The output of DQAM is H', a private estimate of H, which preserves the distribution of original histogram. In first stage, the histogram is partitioned into *disjoint* and *uniform* regions and the trajectory density of each region is computed. Intuitively, partitioning the histogram space into uniform regions improves the signal to noise ratio, which in turn improves the accuracy of the estimation process in stage 2. Instead of estimating the density of every single cell, the algorithm estimates the density of a region and assumes the same density for all cells in the region. Additionally, maintaining the density of a uniform region is significantly easier than a non-uniform region. Since the trajectory data sets are usually sparse, there are potentially many regions with (semi-)uniform distribution in the data. Utilizing this property, the estimation algorithm considers a uniform region as a single entity and uses its density in the private estimation process. Furthermore, the requirement that generated partitions should be disjoint, ensures that each region can be treated independently.

Any algorithm that efficiently generates disjoint and uniform partitions can be used in this stage of DQAM. One approach is to generate all possible partitions and compute the uniformity cost of each partition. The partitioning algorithm should find a set of these partitions, which are

disjoint, cover the entire space, and have the minimum uniformity cost. However, there is no closed-form expression for computing all possible partitions in a $m \times n$ space and the computational complexity grows with the order of $O(2^{m \times n})$. Furthermore, finding the set with the minimum cost is computationally expensive. Another approach is using space partitioning techniques such as quadtree, kd-tree and R-tree. However, the R-tree and kd-tree cannot adapt well to identify disjoint and uniform regions of trajectories. An R-tree cannot cover areas with no trajectory, and the output partitions may be overlapped. A kd-tree requires an ordering among trajectories to do partitioning while neither trajectories nor their counts in the histogram have any order. In addition, the partitioning in kd-tree is based on the density of regions while the outputs might not have a uniform distribution. In contrast, a quadtree decomposes the space into disjoint regions. By defining a partitioning cost to measure the uniformity of regions, a quadtree can partition the space into uniform regions. Whilst, the generated partitions may not be optimal concerning the uniformity cost, the quadtree is computationally efficient, and the final partitioning provides an accurate estimation of uniform regions. However, the regions in a quadtree branch may overlap which could be used to refine parts of a user's trajectory. We need to consider this correlation in the variance of added noise to ensure differential privacy.

Note that regardless of the approach used in stage one, only the outputs (i.e., final partitioning) are used for the next stage of DQAM. Thus, we need to ensure that the final partitions are differentially private as the rest of the generated partitions during the process will be removed. For example, in a constructed quadtree, we need to ensure the privacy of leaves.

In the second stage, DQAM uses the computed final partitions and densities, in addition to the given query set Q, to estimate the distribution of original histogram H. The output is an ε-differentially private histogram H' that is generated using the data and query properties, i.e., DQAM is data- and query-aware. Since the two stages iteratively access the original histogram, we split the ε budget. We will show that we need in total four privacy budgets $\varepsilon_i, i \in \{1,\ldots,4\}$ with $\sum_{i=1}^{4} \varepsilon_i = \varepsilon$. We conducted a pre-study that investigated the impact of different weightings but found no significant impact on the accuracy/utility and hence, set $\varepsilon_i = \frac{\varepsilon}{4}$.

Step 1: Private Partitioning. We develop an efficient algorithm based on the two approaches described above that defines the uniformity as a cost function and partitions the histogram into uniform regions. Intuitively, in a uniform region, all values can be manipulated/treated equally,

D	Trajectory data set, $D \subseteq \mathscr{T}$ and $\|D\| = n$
H	Spatial histogram related to D, $H = \{F, E\}$, $H \subseteq \mathscr{H}$
E	The horizontal and vertical edges in H, $E = \{E_v, E_h\}$
j	Index of a cell in histogram, $j = (row, col)$
$H[j]$	A cell in H with a face $f \in F$ and four edges $e \in E$
f_j	The face component of cell $H[j]$
e_{ij}	The incident edge of two adjacent cells $H[i]$ and $H[j]$
H'	Synthetic spatial histogram
H''	Intermediate synthetic histogram in the synthesizing process
T	Total number of iteration in the synthesizing process
Q	Set of queries $q \in Q$ that $q = \{q_F, q_E\}$
$q[j]$	The jth element of query, $q[j] \in \{0, 1\}$
$q(H)$	Computing the query q on histogram H
P	Set of partitions $p_i \in P$
$p_i[j]$	The jth cell in the region p_i
B	Set of partition densities $b_i \in B$, $b_i = \sum_j c(p_i[j]) / \|D\|$
δ	The threshold of uniformity cost in partitioning (Algorithm 5)
$werr_i$	Error of selected query at iteration i (Algorithm 6)
$c(x)$	The count of x
$\|x\|$	Total number of elements/cells in x

Table 4.1: Summary of notation.

which significantly improves the accuracy of the synthesizing process (see Section 4.2.1). It generates a list of uniform regions P (i.e., the quadtree leaves) representing disjoint partitions of the histogram. The outputs of this stage are the partitions (P) and their densities (B), i.e., the number of trajectories in a region relative to the total number of trajectories in the entire histogram. Since the partitioning process and computing densities need interacting with the original histogram, we use the Laplace mechanism [7] to add noise to them.

Step 2: Private Synthesizing Process. In this step, our algorithm takes the partitions P, densities B and the given range queries Q to privately learn the distribution of the original spatial histogram H and generate an ε-differentially private histogram H'. The learning process is iterative and starts with a uniform trajectory distribution. In each iteration, one query $q \in Q$ is selected. Based on the query error on H', the estimation is updated by rescaling up/down the values in H'. The values in a region $p \in P$ are rescaled if p overlaps with q. The density of region p magnifies/reduces the update scale for that region. Previous work on estimating histogram values while ensuring *local sequentiality* used a heuristic strategy to preserve the dependency. This chapter presents an algorithm to compute the optimal estimation for H' while ensuring the dependency.

For the query selection and adding noise to the query answer, the exponential and Laplace mechanisms are applied, respectively.

The settings used for differential privacy and corresponding standard mechanisms are the same as Chapter 3. Table 4.1 summarizes the notations in this chapter which follows notations used in Chapter 3. The following example is a sample execution of DQAM.

EXAMPLE 1. *An overview of DQAM is presented in Fig. 4.1. The inputs are a histogram H with 4 trajectories and the set of range queries Q shown graphically as rectangular areas over the histogram. A possible output of partitioning can be $P = \{p_1, ..., p_7\}$. This need not to be optimal as defined in Section 4.2 because the partitioning is randomized. The density of partitions are $B = \{b_1 = 2/4 = 0.5, b_2 = 1/4 = 0.25, b_3 = 1/4 = 0.25, b_4 = 1/4 = 0.25, b_5 = 3/4 = 0.75, b_6 = 1/4 = 0.25, b_7 = 1/4 = 0.25\}$. The P and B in addition to Q are then used in step 2 to generate H'.*

4.2 Step 1: Private Partitioning

The first step of DQAM partitions space into uniform regions P and computes their densities B. This stage is independent of the query set and finds P and B as two parameters describing data properties. These parameters are later used in the synthesizing process (step two) to estimate H' as close as possible to H without violating ε-differential privacy. ε_1 is used for computing the cost of a partition and ε_2 for injecting Laplace noise to the partitions densities, where $\varepsilon_1 + \varepsilon_2 = \varepsilon/2$. Algorithm 5 describes our private partitioning and is discussed below.

4.2.1 Quadtree Search

As discussed earlier, the goal is to find partitions in the histogram that are *disjoint* and *uniform*. These requirements are needed to minimize the error in the update step (see s2 in Algorithm 6) where we scale all counts in a region with the same magnitude. To achieve this goal, we propose an approach which is a combination of the two approaches described earlier in Section 4.1. In particular, instead of generating all possible partitions in the histogram, we approximate them with all partitions that could be generated if the histogram was decomposed using a quadtree. Whilst it might come at the expense of a small cost of utility, it leads to a highly efficient algorithm:

Algorithm 5 Private partitioning using a quadtree.

Input: Spatial histogram H,
Set of partitions $P = []$,
Set of partition densities $B = []$,
Deviation cost budget ε_1,
Density budget ε_2

Output: Set of partitions P,
Set of partition densities B

if $|H| = 1$ **then**
  $P.add(H)$
  $B.add(density(H) + \nu')$ # $\nu' \sim Lap(\frac{1}{\varepsilon_2})$
else
  $pcost \leftarrow cost(H) + \nu$ # $\nu \sim Lap(\frac{\Delta cost}{\varepsilon_1})$
  $children \leftarrow split(H)$
  $chcost \leftarrow 1/4 * \sum_{i=1}^{4} cost(children[i]) + \nu_i$ # $\nu \sim Lap(\frac{\Delta cost}{\varepsilon_1})$
  if $pcost - chcost \leq \delta$ **then**
    $P.add(H)$
    $B.add(density(H) + \nu')$ # $\nu' \sim Lap(\frac{1}{\varepsilon_2})$
  else
    for $child \in children$ **do**
      return Partition(child, P, B, ε_1, ε_2)
    end for
  end if
end if
return P, B

(1) Given the partitions in different levels of a quadtree, we can compute the uniformity cost of each partition and add noise to ensure differential privacy; (2) Next, a search procedure expands a quadtree structure can efficiently evaluate all possible combinations and find the least cost set of partitions that are disjoint and cover the entire space of histogram.

The procedure shown in Algorithm 5 combines the two stages. It takes H as input, and returns the leaves of a (potentially unbalanced) quadtree $P = \{p_1, ..., p_k\}$ with their densities $B = \{b_1, ..., b_k\}$. Using the full histogram H as the root node, the algorithm recursively splits H into 4 regions with a size of power of two using the *split()* function. The noisy cost of each region in the current level (*pcost*) is compared to the *average* of noisy cost of its four children (*chcost*). If the difference between *chcost* and *pcost* is greater than a threshold δ, the partitioning proceeds to the

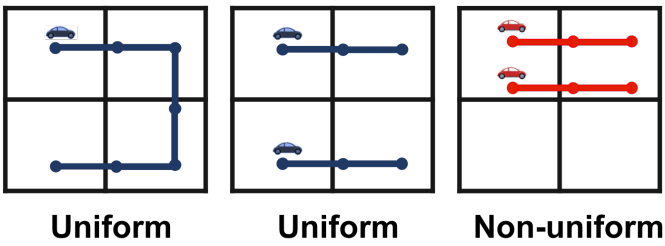


Figure 4.2: Examples of uniform and non-uniform regions.

next level. Otherwise, the node is labeled as "leaf". Since computing the costs require interacting with the original histogram, a random value $\nu \sim Lap(\frac{\Delta cost}{\varepsilon_1})$ is added to each region's cost. The algorithm terminates when all regions are labeled as "leaf", or the size of each region is 1 (one cell). Intuitively, when the node has a good level of uniformity, the children would not significantly improve the uniformity cost.

We define the partitioning cost function to measure the uniformity of a region. Intuitively, a high trajectory density in a part of region incurs a high cost and triggers the algorithm to split the region into smaller ones. Let j denote the cell index in the histogram H and region p be a subset of cell indexes of H. We say a region p is uniform when

$$c(H[j])_{j\in p} = \frac{\sum_{i\in p} c(H[i])}{|p|},$$

where $|\cdot|$ and $c(\cdot)$ represent the set size and the cell count, respectively. Fig. 4.2 shows examples of uniform and non-uniform regions. With this observation, the cost of a region p, denoted by *cost(p)*, is the amount that p deviates from being perfectly uniform:

$$cost(p) = \sum_{j\in p} | \, c(H[j]) - \frac{\sum_{i\in p} c(H[i])}{|p|} \, | \, .$$

4.2.2 Computing Densities

After privately identifying uniform regions $P = \{p_1, ..., p_k\}$, we need to compute the densities of regions $B = \{b_1, ..., b_k\}$. The density of region $p_i \in P$, named b_i defines as the number of trajectories in p_i relative to the total number trajectories in the histogram. Given n, the total number of

trajectories in histogram, the density b_i of region p_i is:

$$b_i = \frac{\sum_{j \in p_i} c(H[j])}{n}.$$

The function $density(H)$ in Algorithm 5 denotes computing the density of regions in the histogram H.

In the second step of *DQAM*, a region specifies the area to be updated, and its density describes the update magnitude.

4.2.3 Formal Guarantees

We analyze Algorithm 5 for three key aspects: privacy, accuracy and efficiency.

Privacy. We show that step 1 is $\frac{\varepsilon}{2}$-differentially private.

Theorem 4.1. *Algorithm 5 satisfies $\frac{\varepsilon}{2}$-differential privacy.*

Proof. Computing the cost of a region that deviates from a uniform distribution, requires interactions with the histogram, and we must add sufficient noise to ensure differential privacy. We use the Laplace mechanism with ε_1 to add random noise to the deviation cost. Note that the quadtree is just used as a search procedure to find a least-cost set of regions and we only reveal the leaves of quadtree (the uniform regions) for step 2 and the other regions are removed. Since there is no overlap among the leaves, we can use ε_1 budget to compute the noisy value of uniformity cost for each region. Hence, the scale of noise is proportional to the sensitivity of regions, denoted by $\Delta cost$, instead of $O(4^{h+1})$ for the entire quadtree. To ensure privacy, we add a random $\nu_i \sim Lap(\Delta cost / \varepsilon_1)$ to the deviation cost of each region. Thus, the total cost of a region is defined in terms of the deviation cost and the error due to the perturbation noise:

$$cost(p) = cost(p) + \nu.$$

$\Delta cost$ shows the maximum change in a region's cost when a trajectory is added to (or removed from) the data set. Assuming k as the maximum length of a trajectory, in the worst case, a trajectory increases the count of k cells in a region p, and we denote the k cells by set p'. We show $\Delta cost \leq 2k$.

Let $cost'(p)$ be the region deviation after adding the trajectory. Assuming that the trajectory only affects the partition p and $|p| \geq k$, we have:

$$cost'(p) = \sum_{j \in p'} | c(H[j]) + 1 - \frac{\sum_{j \in p} c(H[j]) + k}{|p|} | \tag{4.1}$$

$$+ \sum_{j \in p - p'} | c(H[j]) - \frac{\sum_{j \in p} c(H[j]) + k}{|p|} | \tag{4.2}$$

$$\leq k + \sum_{j \in p} | c(H[j]) - \frac{\sum_{j \in p} c(H[j]) + k}{|p|} | \tag{4.3}$$

$$= 2k + \sum_{j \in p} | c(H[j]) - \frac{\sum_{j \in p} c(H[j])}{|p|} | \tag{4.4}$$

Line (3) is derived using the triangle inequality and line (4) follows from $k > 0$. Using this result we have:

$$\Delta cost = cost'(p) - cost(p) \leq 2k.$$

Since a trajectory contributes to every level of the quadtree and thus would be counted multiple times in the overall trajectory count, we divide each trajectory contribution by the number of cells in which it is counted, i.e., its length. Using this approach, we reduce k to 1, i.e., its actual contribution to the overall trajectory count. This implies that $\Delta cost \leq 2$. We use the technique for computing the cost of regions.

In addition to the regions costs, in Algorithm 5 we compute the density of regions located in the leaves of the quadtree. Since computing the density requires accessing the original histogram H, we use the Laplace mechanism with the remaining privacy budget for step 1, ε_2, to add $\nu' \sim Lap(\frac{1}{\varepsilon_2})$ noise to the density count for each region:

$$density(p) = \frac{\sum_{j \in p} c(H[j])}{n} + \nu'.$$

The sensitivity of the density computation is ≤ 1 as it follows the sensitivity of range queries. Computing the region costs is $(\frac{\varepsilon_1}{\Delta cost})$-differentially private and the density calculation is (ε_2)-differentially private. Given $\varepsilon_1 = \varepsilon_2 = \frac{\varepsilon}{4}$ and using serial composition for differential privacy, Algorithm 5 is $(\frac{\varepsilon}{4\Delta cost} + \frac{\varepsilon}{4}) \leq \frac{\varepsilon}{2}$-differentially private. □

Accuracy. The accuracy is measured in terms of the difference between the resultant partitions

and the true partitions for the given histogram.

Theorem 4.2. *With probability at least* $1-\alpha$*, Algorithm 5 generates partitions with cost at most* $TRU+\rho$*, where* TRU *is the cost of true partitions and* $\rho = 16nlog(n/\alpha)/\varepsilon$*.*

Proof. Let TRU be the cost of true partitions and L_ρ be the set of all possible partitions that for $\mathbf{P} \in L_\rho$ the total cost of partitioning is $cost(\mathbf{P}) \leq TRU+\rho$. $\tilde{L_\alpha}$ represents the complement of set L_α. Let ν be the Laplace random noise with scale $b = \Delta cost/\varepsilon_1$ added to each region $p \in \mathbf{P}$. Given $|H| = n$ as the number of cells in the histogram, if $\|\nu\| < \frac{\rho}{2n}$, the total noise added to the partition cost is $n.\frac{\rho}{2n} = \frac{\rho}{2}$ because a partitioning can have at most n regions. This means with such noise still no partitioning in $\tilde{L_\alpha}$ will be generated. Given Algorithm 5 as $\mathscr{A}$, we have:

$$
\begin{aligned}
P(\mathscr{A} \in L_\rho) \geq & \quad P(\nu < \frac{\rho}{2n} \forall p \in \mathbf{P}) && (4.5)\\
= & \quad 1 - P(\exists p \in \mathbf{P}, that, \nu \geq \frac{\rho}{2n}) && (4.6)\\
\geq & \quad 1 - |\mathbf{P}| P(\nu \geq \frac{\rho}{2n}) && (4.7)\\
= & \quad 1 - |\mathbf{P}| exp(-\frac{\rho}{2bn}) && (4.8)\\
= & \quad 1-\alpha && (4.9)
\end{aligned}
$$

The line (7) is achieved by union bound. The line (8) follows the definition of Exponential mechanism with scale b because the absolute value of a random variable from Laplace distribution has an exponential distribution. Now, we can compute the $\rho \geq 2.b.n.log(n/\alpha) = \frac{4.n.log(n/\alpha)}{\varepsilon_1} = \frac{16.n.log(n/\alpha)}{\varepsilon}$ □

Efficiency. Computing optimal partitions in a 2D space are computationally expensive. The complexity of a naive approach is $\Omega(n^2)$ where n is the number of cells in the histogram. Using the quadtree improves the computational complexity of partitioning to $O(nlog(n))$ as: (1) we partition the data space of a histogram with a fixed cell size and fixed counts, and (2) the structure is compatible with the trajectory data sets and each branch proceeds independently. While the achieved quadtree may not be balanced, the maximum height of the tree is of order $log(n)$ due to (1).

4.3 Step 2: Private Synthesizing Process

This section describes the second step of DQAM. The inputs are a set of queries Q, and the partitions $P = \{p_1, ..., p_k\}$ and densities $B = \{b_1, ..., b_k\}$ computed for the histogram H in Section 4.2. In this step, we aim to generate a synthetic histogram H' to be as close as possible to H. We believe this is the first method proposed that is both data- and query-aware for estimating an optimal spatial histogram H'.

4.3.1 Synthesizing Method

Our approach is based on the Multiplicative Weights Update Rule (*MWUR*) [46], [74], which is a query-aware method for estimating the distribution of histogram values. Starting from an initial uniform distribution, *MWUR* iteratively selects one query and updates the estimation by rescaling the values based on the query error. Intuitively, if we improve the estimated data for a query with the largest error, the error of other queries overlapping with this query will also be reduced. Hence, instead of considering all the given queries, *MWUR* only considers a small subset of queries to estimate the distribution of data with high utility for all queries. Based on this insight, we create a *data- and query- aware update rule* for privately synthesizing spatial histograms.

As *MWUR* is query-aware but not data-aware, we have to modify *MWUR* when synthesizing spatial histograms. Providing a data-aware approach incurs three challenges:

1. The cell counts in a region need to be updated/rescaled proportional to the density of that region. However, *MWUR* considers an equal weight/scale for all cells. Instead of assigning actual counts to each histogram cell, *MWUR* uses the relative proportion of counts (referred to as weight) in each cell.
2. *MWUR* only rescales the cell counts contributing to the selected queries. This rescaling strategy is problematic when regions partially overlap with the query area, and some of their cells are outside the query area. For such regions, *MWUR* only rescales the cells inside the query area and leaves the cells outside the query area unchanged. Thus, *MWUR* may degrade the utility of such regions especially those with uniform distribution when answering range queries.
3. The update function in *MWUR* cannot guarantee the *local sequentiality* of cells, which is a

Algorithm 6 Data and query adaptive estimation of H.

Input: Spatial histogram H,
Query set Q,
Set of partitions P,
Set of partition densities B,
Number of iterations T,
Exponential mechanism budget ε_3,
Laplace mechanism budget ε_4

Output: Synthetic spatial histogram H'

Let n be the total number of trajectories in $H \subseteq \mathscr{H}$.
Let H'_0 be a uniform estimation over $\mathscr{H}$.
for $i \in T$ **do**
s1. Query Selection: Select $q_i \in Q$ using Exponential mechanism with budget $\frac{\varepsilon_3}{T}$ and the score function

$$s(q_i, H) = |q_i(H) - q_i(H'_i)|.$$

s2. Update: Given the error of q_i as $werr_i$ perturbed by a noise from $Lap(T/\varepsilon_4)$, rescale the count of cell j in region $p \in P$ with density $b_p \in B$ as

$$H'_{i+1}[j] \propto H'_i[j].exp(q_i[j].werr_i.b_p/2n).$$

s3. Optimal Estimation: If the update *scaled down* the counts and *inconsistency happened*, compute the nearest histogram as H' that ensures the consistency of cell counts.
end for
return H'

key property of spatial histograms. Ignoring this property can lead to an *inconsistent* output and in turn, reduces the utility.

In Section 4.3.2 we address the first and second challenge by proposing a data- and query-aware update function. Given the uniform regions and their densities from step 1, our update function assigns a scaling factor to each region proportional to its density and preserves the uniformity of the updated regions. To address the third challenge, we need to ensure the *consistency* of the updated spatial histogram. *A consistent spatial histogram guarantees that the answer of a range query monotonically increases with its query area when the larger range query includes smaller range queries.* Chapter 3 developed an efficient approach to generate consistent output but the resulting histogram may not be optimal. In Section 4.3.3 we discuss consistency for spatial

histograms and develop a set of requirements to ensure consistency. We propose a highly efficient method to compute the optimal spatial histogram while ensuring consistency. The privacy budgets ε_3 and ε_4 are used for this step to ensure differential privacy. Algorithm 6 shows our data- and query-adaptive solution for estimating the true histogram.

4.3.2 Update Function

Algorithm 6 shows, in each iteration $i \in T$, the Exponential mechanism consumes $\frac{T}{\varepsilon_3}$ budget to privately select a query $q \in Q$ such that q has worst error on the current estimation H'_i. The error of selected query q is then used to update the estimation, i.e.,

$$werr_i = (q(H) + \nu_i) - q(H'_i).$$

$\nu_i \sim Lap(\frac{T}{\varepsilon_4})$ is injected noise to the true answer of q.

In each iteration, *MWUR* reduces the estimation error by rescaling the weight of cell counts. Each cell count $H'_i[j]$ intersecting the query area is multiplied by a scaling factor s_i computed as:

$$s_i \propto exp(q_i[j].werr_i).$$

$q_i[j]$ is j^{th} element value of query q selected in iteration i. $q_i[j] = 1$ if the cell with index j is *inside the query area* and $q_i[j] = 0$ otherwise. The scaling factor (i) only rescales the count of *cells inside a query area* and (ii) all the counts are rescaled equally. Such scaling function requires the original histogram H to be uniformly distributed and fails if the densities in different regions have non-uniform distribution (see Section 4.2.1).

We propose a scaling function that (i) updates *all cells in a region* if the region intersects the query area and (ii) rescales a cell count proportional to the density of region that the cell belongs to it. For a region $p \in P$ with density $b_p \in B$, our update function rescales the cell $p[j]$ with the following factor:

$$s_i \propto exp(q_i[j].werr_i.b_p),$$

where $q_i[j] = 1$ if the region p intersects q and $q_i[j] = 0$ otherwise. Note now the affected area by $q_i[j]$ is expanded beyond the query region. The query error $werr_i$ determines the magnitude

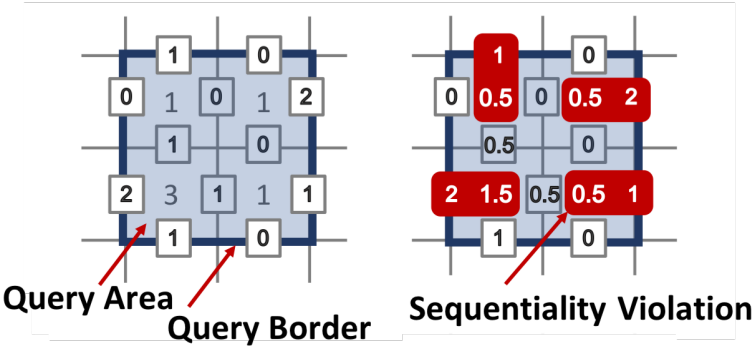


Figure 4.3: The effective area of a range query and the query borders; the local sequentiality violations caused after updating the edge/face values in the query area by a scaling factor of 0.5.

and direction of update with respect to the selected query. b_p amplifies the magnitude of update regarding the region density and $q_i[j]$ identifies the extent of update which is the union of query area and the regions intersected the query. According to the scaling factor, we define our update rule for iteration $i \in T$ as follows:

$$H'_{i+1}[j] \propto H'_i[j].exp(q_i[j].werr_i.b_p/2n), \tag{4.10}$$

where n is the total number of trajectories in the histogram.

4.3.3 Optimal Estimation

For a spatial histogram, it is key to preserve local sequentiality of cells, which means any change of a cell value cannot be performed independently without taking into account the adjacent cell values. As mentioned before, the edge values in spatial histograms capture the local sequentiality: an increase in a cell also leads to an increase in an adjacent cell if the edge value indicates that the represented trajectory (or more generally object) spans over both cells. Hence, local sequentiality imposes a *constraint* between each pair of adjacent faces f_i and f_j and their incident edge e_{ij}, which has to comply with the constraints $e_{ij} \leq f_i$ and $e_{ij} \leq f_j$ (a trip is always started in a cell per construction and might or might not cross an edge). This constraint is a key requirement for *consistency* in a spatial histogram as any change of data must satisfy it. Violating this constraint can even lead to a negative number of trajectories for a range query on a spatial histogram.

The update rule in Section 4.3.2 does not consider this constraint when rescaling the face/edge

values while computing a synthetic histogram which may lead to inconsistencies. Fig. 4.3 shows an example of the sequentiality violation where the query fully matches to the region. Since the update function rescales the values *inside the query area* it causes violations in edges placing on *borders of the query*. A heuristic strategy may ensure the constraint by increasing the violated face value to the maximum of edge values corresponding to the face. However, such a heuristic strategy can lead to overcorrect and degrade the estimation accuracy. To guarantee the consistency of the synthesized histogram, we describe an approach using linear programming called *consistent inference*. Its aim is to compute the closest consistent histogram to the estimated histogram H' that satisfies the constraints.

Set of Consistency Constraints (C). Every edge count is at most equal to the minimum face count of its adjacent cells:

$$c(e_{ij}) \leq \min\{c(f_i), c(f_j)\}, \forall e_{ij} \in E'', \text{and } \forall f_i, f_j \in F''.$$

Definition 4.1. *Consistent Inference. Let T be the total number of iterations and H' be the estimated histogram in iteration $i \in T$. Given the set of constraints $\mathscr{C}$ between face and edge counts, the* consistent inference *function returns the histogram $H'' = \{F'', E''\}$ that satisfies the constraints in $\mathscr{C}$ while minimizing $\|H'' - H'\|$.*

We formulate the consistent inference problem as follows:

$$\begin{aligned} \underset{H''}{\text{minimize}} & \sum_{i \in H'} |c(H''[i]) - c(H'[i])| \\ \text{s.t. } & c(H''[i]) \geq 0, \forall i \in \{1, ..., |H''|\} \\ & c(e_{ij}) \leq \min\{c(f_i), c(f_j)\}, \forall e_{ij} \in E'', \forall f_i, f_j \in F''. \end{aligned}$$

We use linear programming to solve this problem. Note that a constraint violation can only happen when the update rule *scales down* the counts or the query error in (4.10) is negative. Checking this condition in Algorithm 6 s3 significantly improves the efficiency of this algorithm.

4.3.4 Formal Guarantees

We evaluate Algorithm 6 in three key aspects: privacy, accuracy and efficiency.

Privacy. We show Algorithm 6 is $\frac{\varepsilon}{2}$-differentially private.

Theorem 4.3. *Algorithm 6 satisfies $\frac{\varepsilon}{2}$-differential privacy.*

Proof. Given $\varepsilon_3 = \varepsilon_4 = \frac{\varepsilon}{4}$, the query selection and computing the error of selected query, each uses $\frac{\varepsilon}{4T}$ of the privacy budget. Computing the optimal estimation is a post-processing and hence does not spend any privacy budget. The algorithm makes T calls of the first two stages and according to the serial composition, the budget share accumulates in the calls which result $\frac{\varepsilon}{2}$ privacy level for the algorithm. □

Accuracy. The accuracy is measured in terms of the maximum error bound in answering per $q \in Q$ using the estimated histogram H'. The error is because of *perturbation noise* to ensure differential privacy and *estimation error* in rescaling the estimated counts in H' toward the true counts in H.

We show that the accuracy bound of Algorithm 6 meets the bounds with the state-of-the-art approaches in [46]. However, this is the worst case bound. In our experiments, we show that defining the update rule based on regions and optimizing the estimation with respect to the consistency constraints significantly improves the output accuracy.

Theorem 4.4. *Given a spatial histogram H, for any $Q, T \in \mathbb{N}$, and $\varepsilon > 0$, with probability $\geq 1 - 2T/|Q|$, Algorithm 6 estimates H' such that*

$$\max_{q \in Q} |q(H) - q(H')| \leq 2n\sqrt{\frac{log|H|}{T}} + \frac{10T log|Q|}{\varepsilon}$$

Proof. The proof follows the analyze of accuracy in [46]. We show that our new update rule and later the linear programming do not affect the worst case bound of accuracy. Without further amendments, the proof in [46] carries over from 1D to 2D for the spatial histogram in our work. It first computes a bound on the perturbation noise regarding the query error. Second, it uses the relative entropy to bounds the estimation error imposed by the update rule. Last, it combines the two bounds to achieve the worst case error bound on the accuracy. Our improvements affect the second step of analysis and we show it does not change the bound of estimation error.

Our update rule rescales the count of each cell $x \in H'$ proportional to the density of region that x belongs to that, i.e., $b_x \in B$. After scaling the counts in each round, the optimal estimator multiplies the cell counts by a multiplier α_x where $\alpha_x \geq 0$. Considering a maximum multiplier in each round, we define α as the maximum bound on the multipliers across all rounds, i.e.,

$$\alpha = \max_{i \in T} \max_{x \in H'} \alpha_x.$$

According to the definition of relative entropy, the difference between the H and H' after each update round improves as:

$$\psi_{i-1} - \psi_i = \sum_{x \in H} c(H[x]) log(\frac{c(H'_i[x])}{c(H'_{i-1}[x])})/n, \tag{4.11}$$

where the ratio $c(H'_i[x])/c(H'_{i-1}[x])$ can be written as $\frac{\alpha}{\beta_i} sp_i$ such that $sp_i = exp(q_i[x]\eta_i b_x)$ with $\eta_i = (m_i - q_i(H'_{i-1})/2n$ and $b_x \in B$ is the density of region that the cell x belongs to it. β_i is the re-normalization factor. As the cells may have different density coefficients, we consider $\mathbf{b} = \{\mathbf{b} \in B \mid \forall b_i \in B, \mathbf{b} \geq b_i\}$. Expanding the equation gives:

$$\psi_{i-1} - \psi_i \quad = \quad log(\alpha) + \mathbf{b}\frac{\eta_i}{n} q(H) - log(\beta_i).$$

Using the Taylor expansion of sp_i and $q_i[x]^2 \leq 1$, the β_i bound will be:

$$\begin{aligned}
\beta_i &= \sum_{x \in H'} (\alpha.sp_i.c(H'_{i-1}[x]))/n \\
&\leq \sum_{x \in H'} \frac{\alpha}{n}(1 + \mathbf{b}q_i[x])\eta_i + \mathbf{b}^2\eta_i^2)c(H'_{i-1}[x]) \\
&= \alpha(1 + \frac{\mathbf{b}\eta_i}{n} q_i(H'_{i-1}) + \mathbf{b}^2\eta_i^2).
\end{aligned}$$

Inserting this bound to the equality 4.11 and that $log(1+x) \leq x$ and $\mathbf{b} \leq 1$, cancels $log(\alpha)$ and gives:

$$\begin{aligned}\psi_{i-1}-\psi_i &\geq \mathbf{b}\frac{\eta_i}{n}q(H)-log(1+\mathbf{b}\frac{\eta_i}{n}q_i(H'_{i-1})+\mathbf{b}^2\eta_i^2)\\ &\geq \eta_i(q_i(H)-q_i(H'_{i-1})/n-\eta_i^2.\end{aligned}$$

By introducing the definition of η_i to this result, it gives:

$$\psi_{i-1}-\psi_i \geq (\frac{q_i(H'_{i-1})-q_i(H)}{2n})^2-(\frac{m_i-q_i(H)}{2n})^2,$$

which is the same bound on the estimation error in [46]. The remaining of the proof follows the accuracy analysis in [46]. □

With the fact that $q(H)\in[0,n]$ for all $q\in Q$, we can reduce the bound to less than n by choosing T to be greater than $4log(|H|)$.

Efficiency. The computational cost of Algorithm 6 depends on the three steps in each iteration $i\in T$: query selection, updating the estimation, and computing the optimal estimation. For the query set Q, computing the score of each query $q\in Q$ has $O(|H|)$ cost where $|H|=r\times c$ is the total number of cells in H. Thus, the total cost of query selection is $O(|H||Q|)$. Updating the estimation takes $O(|H|)$ as the cell counts in regions overlapping with the query should be rescaled by the scaling factor. The linear programming solution in the last step has the worst case cost of $O(|H|^{3.5})$ [76]. The three steps are called for T times that results in the complexity of $O(T(|H||Q|+|H|^{3.5}))$ for the algorithm.

In practice, the total number of queries selected through the iterations (one per iteration $i\in T$), call Q', is much less than the $|Q|$, i.e., $|Q'|\ll|Q|$. The T is also chosen to be a small constant such that $T\cong log(|H|)$ due to the accuracy bound above. These two small constants have a negligible effect on the first part of the complexity cost. In addition, using the violation check after an update can significantly improve the complexity of the algorithm as we do not need to call linear programming function.

4.4 Experimental Evaluation

This section evaluates the performance of DQAM in different settings explored by recent works including privacy budgets, query sets, data sets, the entire world size (i.e., (1) number of records in the trajectory data set, and (2) number of cells in the histogram), and running time. We contrast the quality of generated histogram with the output of recently proposed algorithms in terms of accuracy and utility.

4.4.1 Experimental Setup

Considering the number of cells (rows times columns) in a histogram as its resolution, we set the resolution of a histogram as a power of 2. Depending on the speed limit of moving objects, each data set may need a different level of resolution for the histogram. In a data set with $50km/h$ limit and trajectory sampling rate of 3 GPS coordinates/sec, a histogram with resolution 8 aggregates the trajectory counts into $2^8 \times 2^8$ cells with $39m^2$ per cell which is fairly accurate for capturing movements in metropolitan areas. Higher resolutions can more accurately capture the movements but are computationally more expensive. Unless otherwise specified, we map the data sets into histograms of resolution 8. We choose the partitioning threshold $\delta = 4/\varepsilon_1^2$ in Algorithm 5 to cover the variance of total noise added to the costs of a partition and its potential children. This choice of δ prioritizes high uniformity in partitioning which leads to highly uniform partitions. We found the computed results stable with this setting for δ. The number of iterations T is chosen from the set $\{10, 12, 14, 16, 18, 20\}$. For each chosen T value, we run the algorithm 5 times and compute the mean error. Finally, we report the lowest error.

Data sets. One real and one synthetic data set are used in our experiments. The real data set is PORTO-TAXI containing the trajectory of taxi-cabs in the city of Porto, Portugal [73]. It has about 1.7 million trajectories of various lengths from 443 taxi-cabs in one year. PORTO-TAXI is a large data set with a highly skewed distribution of trajectories going toward the city center that mapping it to a spatial histogram, results in a histogram with many sparse areas. It is important for the algorithm to be able to capture the distribution information and utilize it by privately estimating the true distribution. The synthetic data set is called MELB-CAR which is generated by Minnesota Web-based Traffic Generator (MNTG) [75]. We simulated 1,000,000 vehicle trajectories for 20

Mechanism	Data-aware	Query-aware	Greedily Consistent	Optimally Consistent
LM		✓		
MWEM		✓		
DAWA	✓	✓		
PriSH		✓	✓	
DQAM	✓	✓	✓	✓

Table 4.2: DQAM versus existing works in terms of technical and output features.

time units in the city of Melbourne, Australia. MELB-CAR represents a data set with a uniform distribution where the counts are almost equal in all histogram cells. The two data sets are used to measure the effect of data set distribution on the quality of estimation. To show the effect of data set size $|D|$ on the estimation quality, we use three samples of each data set in the experiments, $|D| \in \{1000, 100{,}000, 1{,}000{,}000\}$. We used a square of 100km^2 as the spatial geometry to map the data set into a histogram. Unless otherwise specified, a data set of 1000 trajectories from PORTO-TAXI is used for the experiment. We note such a small data set with non-uniform distribution results in a sparse histogram and provides the most difficult setting for the algorithm to estimate the true distribution of data. Our results are of high quality even in this hard setting.

Queries. We use queries in the experiments that are uniformly sampled from the domain of histogram H. Given $X = [x,x']$ and $Y = [y,y']$ as domains of columns and rows in H respectively, a query is generated as a rectangle $([x_1,x_2],[y_1,y_2])$ where $x_1,x_2 \in X$ and $x_1 \leq x_2$; $y_1,y_2 \in Y$ and $y_1 \leq y_2$. Unless otherwise specified, the default size $|Q| = 16{,}000$ is used for the query set.

Measures. We evaluate the accuracy of estimation in terms of the error in answering queries and measure it using the average L_1 error per query. In addition, we expect the estimated histogram have high utility in answering arbitrary queries from the class of range queries. The utility of estimated histogram H' is evaluated as the difference between its distribution H'/n and the distribution of original histogram H/n:

$$RE(H||H') = \sum_{j \in H} c(H[j]) log(\frac{c(H[j])}{c(H'[j])})/n,$$

where n is the total number of trajectories in the given data set. This measure is also known as Kullback-Leibler Divergence (KLD) and is used in prior work [46] to evaluate the utility of estimation. This enables us to compare our approach to previous work.

Baselines. We compare the performance of DQAM with four existing works: Laplace mech-

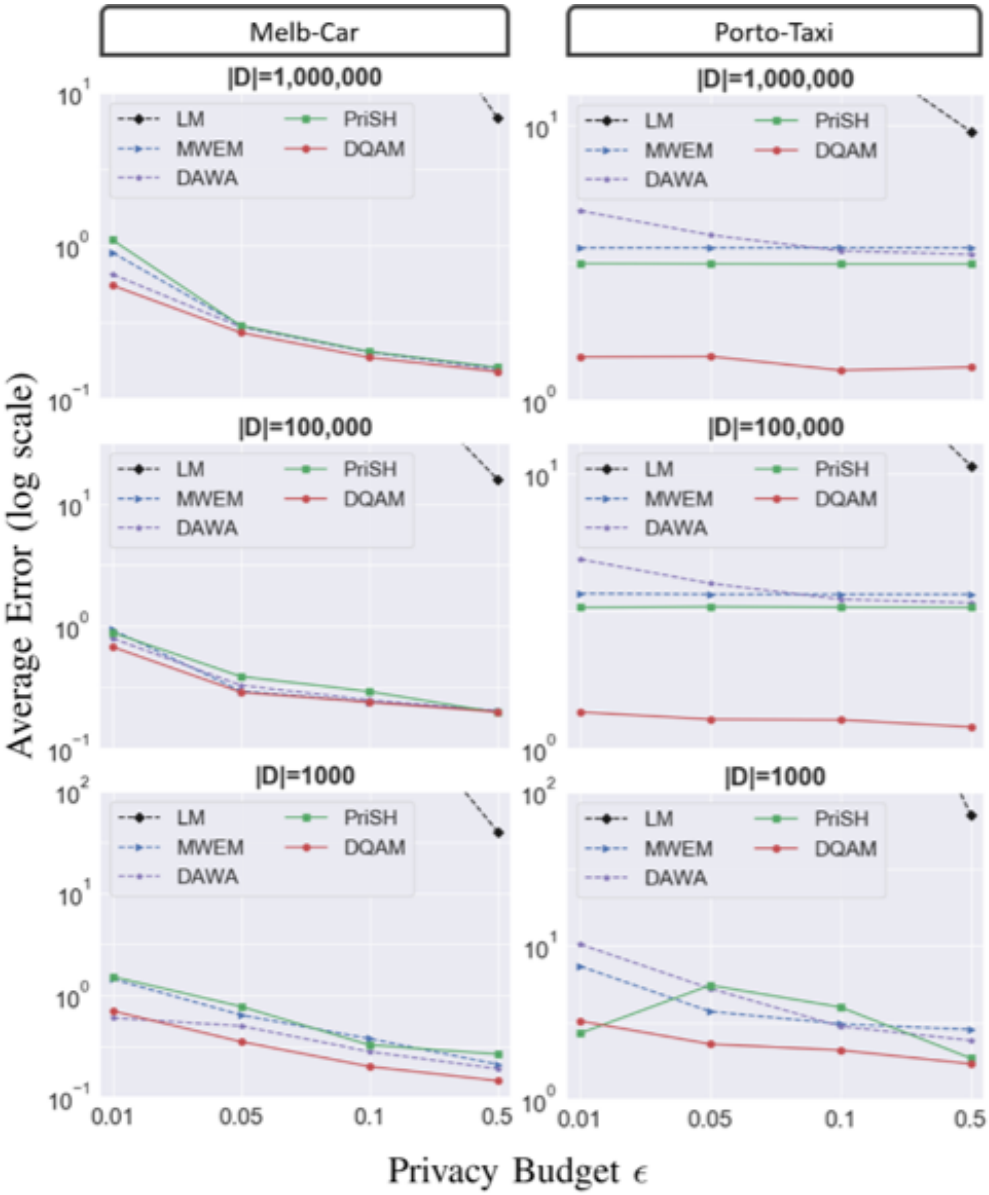


Figure 4.4: The effect of data distribution and size on the error. Dashed line means the output may not have consistency.

anism (LM) [7] as a naive approach, MWEM [46], DAWA [51] and PriSH (Chapter 3). Table 4.2 contrasts the baselines and DQAM in terms of technical features, i.e., (1) utilizing data properties (data-aware), (2) utilizing query properties (query-aware), and the output features, i.e., (3) preserving consistency or local sequentiality (greedily consistent) as well as (4) Optimizing the output while ensuring consistency (optimally consistent). Measures (1) and (2) ensure that the mechanism can accurately learn the distribution of data and maintain the data properties. Measure (3) guarantees the validity of generated trajectories and measure (4) optimizes the utility of output with respect to the sequentiality constraint. As the table shows, Laplace mechanism and MWEM only consider the query properties to identify the scale of noise. Laplace mechanism is a naive approach in which the noise is directly scaled based on the dependency of queries. In trajectory data, the sensitivity is equivalent to the number of queries which is considerably large. MWEM, however, selects a few numbers of queries which significantly reduces the scale of added noise. DAWA, is data-aware and query-aware similar to DQAM but does not consider the local sequentiality in trajectories (see Section 4.5 for more details). Hence, DAWA fails to preserve consistency in the output. On the other hand, PriSH maintains the consistency of data. However, it only takes advantage of query properties in scaling the noise and generating the output histogram. Additionally, PriSH fails to generate the optimal histogram with respect to the consistency requirements (see Section 4.5 for more details). DQAM is the only mechanism that is data- and query-aware, and generates a histogram that is optimal with respect to consistency preservation.

Since MWEM and DAWA are designed for 1D histograms with no local sequentiality, we only use the face component to evaluate the algorithms. For these algorithms, we convert the 2D histogram to 1D using Hilbert curves. The number of iterations in PriSH and MWEM are chosen as explained for DQAM, and ε is split evenly between the query selection and the update function in the algorithms. According to [51], we consider the ratio $r = 0.5$ for dividing privacy budget ε between the data-aware and query-aware steps in DAWA. In our evaluations, we do not report results of DQAM and PriSH algorithms without consistency. Note that Laplace mechanism , MWEM and DAWA do not guarantee the consistency of generated histogram. Since we only consider the face component to evaluate these mechanisms no penalty is applied for the inconsistency of output. The corresponding results of Laplace mechanism , MWEM and DAWA are depicted by dashed lines in graphs to indicate this limitation.

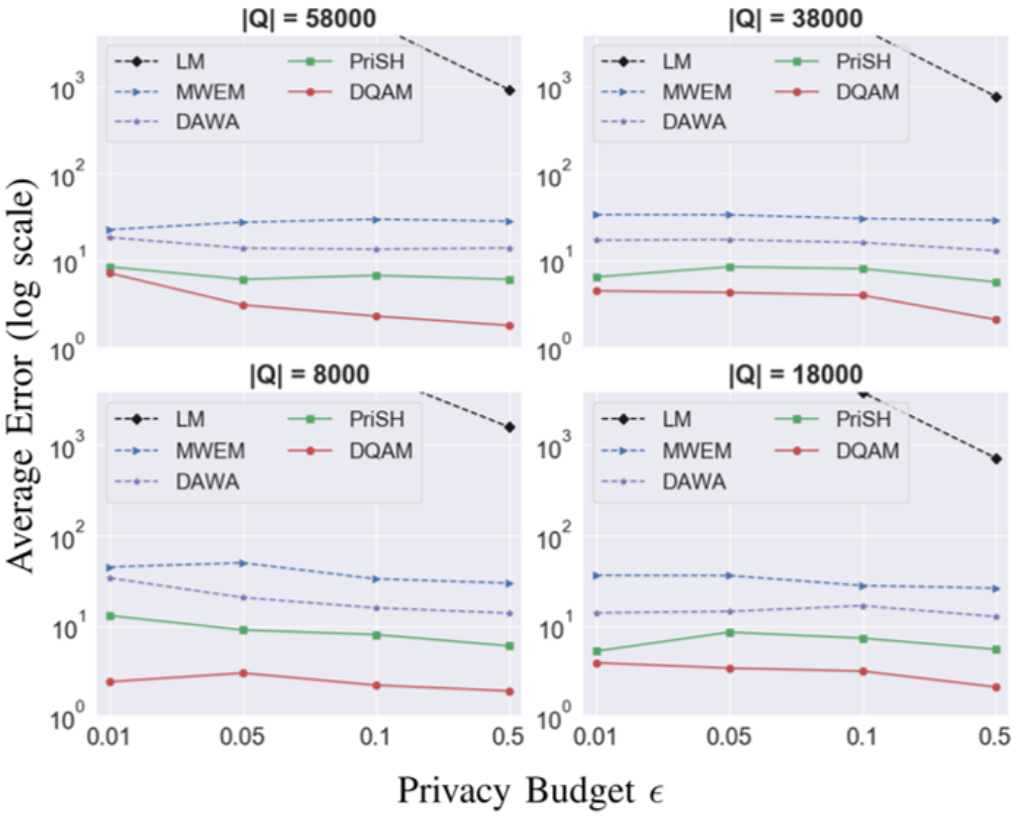


Figure 4.5: The effect of query set size on the estimation error.

ε	LM	MWEM	DAWA	PriSH	DQAM
0.01	500.86	5.64	3.65	2.50	1.73
0.05	500.15	4.48	2.00	2.81	1.54
0.1	429.71	3.37	1.24	1.67	0.53
0.5	285.44	3.10	0.86	0.39	0.27

Table 4.3: Standard deviation of DQAM error and its competing algorithms for each ε on the sample of PORTO-TAXI with 1000 trajectories.

4.4.2 Accuracy Evaluations

Data size and distribution. Fig. 4.4 evaluates the accuracy of DQAM versus existing works in different data set sizes where results are depicted in log scale. For larger datasets, the counts in the corresponding histogram are larger and hence, less sensitive to noise. That is the algorithms usually have higher accuracy for larger data sets. As the results show, DQAM achieves mostly the lowest error specifically for smaller εs. This advantage is more significant on PORTO-TAXI data set where DQAM accurately captures the sparse/dense regions and estimates the counts properly. In contrast, PriSH has an unstable accuracy across different ε values as the heuristic strategy in PriSH fails to efficiently estimate histogram in the presence of noise.MWEM and DAWA work good on uniform distribution in MELB-CAR, but their accuracy considerably decreases on non-uniform distribution. The generated histogram, though, is not valid because of inconsistency in data. The error of generated histogram by Laplace mechanism is always considerably higher than other mechanisms due to naively scaling the added noise. Table 4.3 reports the standard deviation of accuracy for algorithms for different ε values. As expected, the standard deviation of Laplace mechanism is always high, and this naive approach does not show a stable behaviour. DQAM has the least standard deviation. In contrast with PriSH, DQAM employs the optimal consistent estimation resulting in smaller and more stable error which decreases for larger εs. Inconsistency of generated histograms by MWEM and DAWA result in large variations in the error.

Size of query set. As the set of given queries are randomly generated, the queries may have different shapes. This diversity helps the query selection in DQAM to capture the information from the spatial histogram efficiently. We evaluate the effect of changing the query set size on the error of estimated histogram. Fig. 4.5 depicts the error of DQAM versus baseline methods. DQAM often achieves higher accuracy comparing to the baselines. Furthermore, the error is almost stable and slightly increases in larger query sets. This behaviour agrees with the Theorem 4.4 in that the

size of the query set has a logarithmic effect on the worst case error.

Resolution of histogram. Fig. 4.6 investigates the effect of changing the histogram resolution on the quality of estimation in DQAM. As it was expected, higher resolutions cause higher errors. However, DQAM works well in different resolutions comparing to the other algorithms. The significant increment in the error of Laplace mechanism , MWEM and DAWA is due to accumulation of error caused by local sequentiality violations.

As mentioned above, the reported results are the average value of 5 independent runs of an algorithm. The ratio of accuracy improvement achieved by DQAM comparing to the competing algorithms changes in different settings. The highest ratio achieved in our experiments was 7.4 for $|D| = 1000$, $|Q| = 8000$, resolution 8 and $\varepsilon = 0.01$ comparing the average error of DQAM versus PriSH shown in Fig. 4.5: PriSH/DQAM $= 14.0654/1.901 = 7.3989$. While the ratio of DQAM accuracy versus Laplace mechanism , MWEM and DAWA is significantly higher (minimum ≥ 60 for this setting), we did not report it as the output of these algorithms are not consistent due to local sequentiality violations.

4.4.3 Utility Evaluation

The utility of estimated histogram shows its quality in representing the original histogram. Utility is in fact a measure of quality of generated histogram in answering arbitrary range queries, the ones not included in the query set or not selected in the mechanism process. As mentioned before, We measure the utility of estimated histograms by *KLD* which is used in prior works. We use this measure to show employing both data and query information in DQAM has significant effect in improving the algorithm in detecting sparse/dense regions. Fig. 4.7 depicts that DQAM perfectly captures the data distribution properties in both uniform and non-uniform distributions. For clarity in presentation, we focus on DQAM, PriSH and MWEM in this experiment.

4.4.4 Efficiency Evaluation

Fig. 4.8 evaluates the running time of DQAM on small to large samples of PORTO-TAXI and MELB-CAR datasets with $\varepsilon = 0.1$ and a fixed resolution of histogram. The partitioning (step 1 in DQAM) takes less than a second in all settings, and the synthesizing process (step 2 in DQAM)

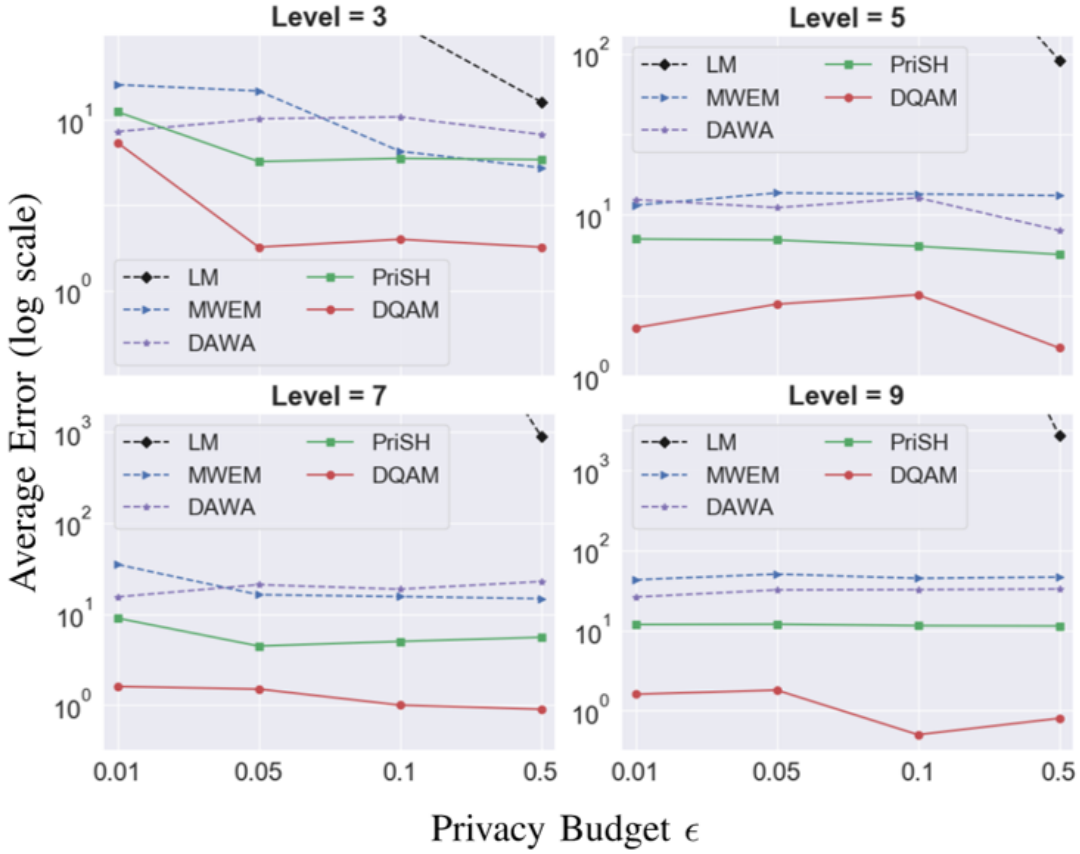


Figure 4.6: The effect of histogram resolution on the estimation error.

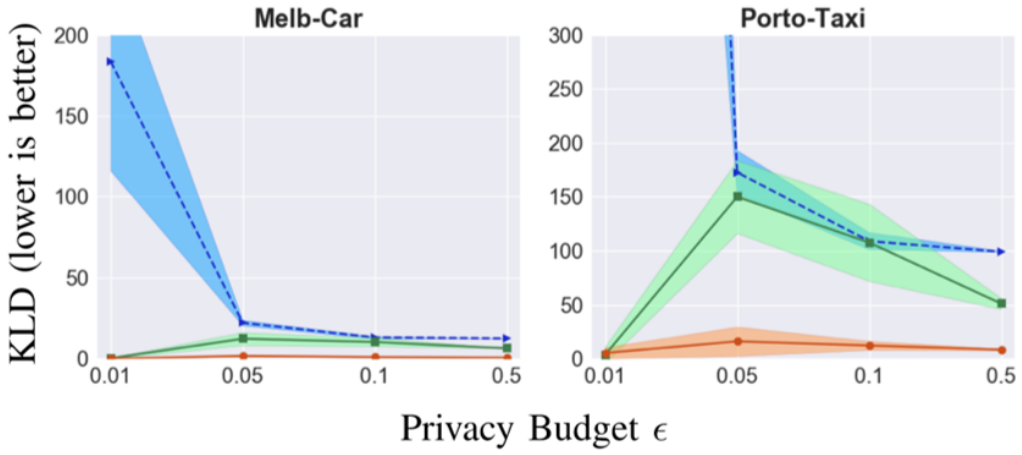


Figure 4.7: The utility of output by DQAM (orange), PriSH (green) and MWEM (blue) algorithms. Lower KLD means higher utility.

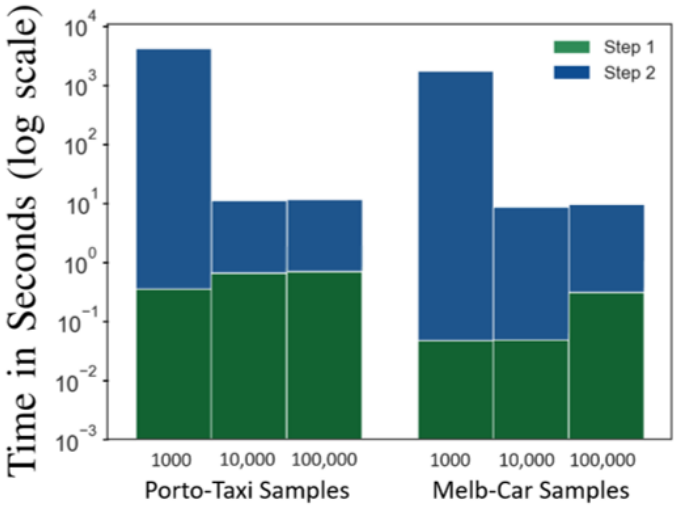


Figure 4.8: Running time of *DQAM* on small to large datasets.

mostly contributes to the running time. The total running time for large samples of PORTO-TAXI and MELB-CAR are ≈ 10 and ≈ 8 seconds, respectively. In small samples where the histogram is very sparse, some updates in the synthesizing process cause consistency violations which, in turn, cause the second step to take more time to compute an optimal histogram satisfying consistency. Note that a violation should be corrected as soon as it happened in the synthesizing process as it can affect the updates in next iterations. Indeed, the accumulation of consistency violations leads the solution to deviate in a wrong direction that it may be far from the optimal point that causes the mechanism to take a long time to compute the optimal estimation in the feasible space of the problem. The total running time for small datasets is ≈ 60 minutes for PORTO-TAXI and ≈ 25 minutes for MELB-CAR. According to the results our mechanism is eminently practical and scalable.

4.5 Discussion

In Chapter 3, we developed a mechanism named Private Spatial Histogram PriSH for range queries on trajectories. PriSH publishes a synthetic spatial histogram under ε-differential privacy. It is a query-aware mechanism that extends the idea of MWEM in [46]. PriSH, takes a spatial histogram and a query set as input and utilizes the correlation between the queries to estimate the distribution of the original histogram privately. To maintain the consistency in the histogram, PriSH

uses a *heuristic* approach that may result in a histogram far from the original spatial histogram. The reason lies in the *heuristic* approach that locally ensures consistency and may lead to over-correction. In this chapter, we proposed a data- and query- aware mechanism that utilizes the trajectories density in different regions as well as the given queries correlation to estimate the optimal spatial histogram with significantly higher utility. Our query-aware strategy employs a linear programming approach to provide a guarantee on optimally consistent histogram which leads to significant improvement in the utility of results.

4.6 Summary

We proposed DQAM, a data and query-aware mechanism for privately publishing spatial histograms and answering range queries on trajectories. The first step of DQAM identifies disjoint uniform regions in a histogram and computes the density of each histogram region. It uses a quadtree as it enables efficient space partitioning. Based on the quadtree, we can measure the uniformity for each step of the partitioning. The partitions and densities are then used in the second step of DQAM for learning the true distribution of histogram. The learning process uses a given query set as well as the partitions and densities from the first step to estimate the true distribution and to improve the quality of estimation in each iteration. Our experiments show that DQAM significantly improves the accuracy over the state of the art methods, specifically on data sets with a non-uniform distribution while ensuring consistency, i.e., local sequentiality. Our experimental results show that we have improved utility by a factor 7.4 (see Fig. 4.5). In the future, we will focus on differentially private trajectory representations that retain high data utility and thus can be used for any spatial query type as current techniques cannot achieve the required data utility.

Chapter 5

TGM: A Generative Mechanism for Publishing Trajectories with Differential Privacy

In this chapter, we describe a new algorithm called Trajectory Generative Mechanism (TGM) for publishing trajectory datasets *with ε-differential privacy guarantee which achieves substantially higher computational efficiency and utility (practical) than the state-of-the-art algorithms. Our algorithm first* encodes *(models) the data as a graphical generative model and accurately captures the statistics of moving object trajectories. Using this model, TGM then privately* generates *synthetic trajectories such that the noise is optimally added to capture the movement direction of an object. Our algorithm (1) preserves both the spatial and temporal information of trajectories in the generated dataset, (2) requires less memory and computation than competing approaches and (3) preserves the properties of real trajectory data in terms of travelled distance and stay location. We demonstrate the performance of TGM on both real and simulated datasets with a wide range of settings. Our experimental results show that TGM achieves high utility and efficiency by using the properties of the data.*

5.1 Introduction

WITH the ubiquity of smart devices, location service providers are able to collect huge number of location points about individuals' movements with very high accuracy. Such widely covered data enables us to analyze people's daily habits, such as activity sequences and mobility patterns. Upcoming location-based services in intelligent transportation systems and urban planning such as Connected and Autonomous Vehicles and Mobility-as-a-Service are instances of Internet of Things (IoT) and require detailed trajectory data about individuals to provide high

quality of services. For example, autonomous vehicles contain sensors of diverse types as per IEEE P2413 [77] (e.g. camera, object detector, light detector) that trace the vehicle's location for safety and customized services. Smart buildings are another instance various sensor types (e.g., cameras, cell phones, WiFi access points) that track individual's location and activity to provide customized experience based on the user's context. However, the fine-grained collected data of locations are used by an increasing number of service providers raising deep privacy concerns [78], [79]. Providers are able to infer an individual's sensitive locations (e.g., home) or their stay locations (e.g., clinic) where people may spend substantial periods of time. Many studies have shown that the privacy threats of releasing the trajectory datasets are significantly high [5]. As the spatio-temporal trajectories act as pseudo-identifiers [57] that can reveal an individual's identity and their sensitive information. The significance of this threat is highlighted by Montjoye et al. [57]: they show only 4 points in a trajectory are sufficient to uniquely identify 95% of individuals in a large dataset. This problem is more severe for longer trajectories as their movement pattern can be significantly different from the general movement of trajectories in the dataset and easier to be distinguished.

Recent studies have developed ε-*differentially private* mechanisms for publishing trajectory datasets [18] [19] [20] [21] with strong privacy guarantees [7]. ε refers to the privacy level where a lower value provides a stronger privacy guarantee. The main advantage of publishing the dataset rather than aggregated information of data is that it is not limited to particular set of queries for analysis. The key challenge, however, in developing a differentially private mechanism is to preserve an accurate trajectory pattern when adding noise to ensure privacy. A naive mechanism that scales the noise proportional to the length (i.e., number of locations) of a trajectory can adversely diminish its utility. Existing mechanisms first pre-process trajectories by mapping them to a grid, then extract the most frequent patterns from these coarsened trajectories (grid cells), afterwards build a probabilistic model using the patterns with added noise, and finally post-process the noisy model to make the trajectories consistent. This model is then used to generate synthetic trajectories.

However, the proposed mechanisms fail to accurately preserve the trajectory patterns in their generated data. The pre- and post-processing steps in those mechanisms can distort the trajectory properties such as travelled distance and frequent trajectory patterns. In addition, the efficiency of

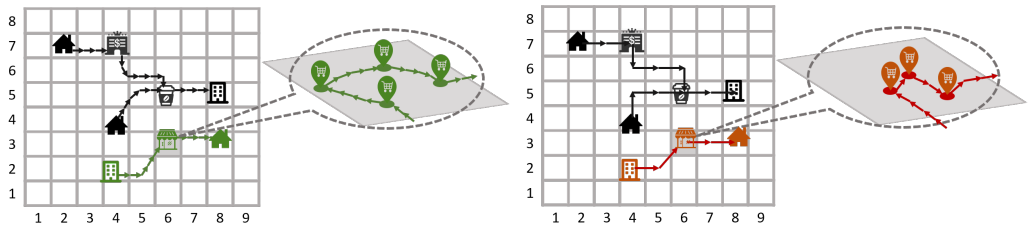
existing mechanisms is highly sensitive to the size of the corresponding spatial space. The running time and the memory required increases by an exponential magnitude when the size of cells in the corresponding grid decreases (i.e., higher resolution). For example, the state-of-the-art mechanism DPT (Differentially Private Trajectories) [20] fails to accurately preserve the travelled distance and frequent trajectory patterns even in a medium-sized metropolitan area such as [73]. An established measure for utility in maintaining travelled distanced is the Jensen-Shannon divergence (JSD) with a value in the range $[0,1]$ where a lower JSD indicates a higher utility. Given the distribution of travelled distances in the original dataset, JSD is used to compute its distance with the distribution travelled distances in the generated trajectories (see Section 5.6). Even for a moderate privacy budget of $\varepsilon = 0.5$ DPT only achieves 0.268 JSD error, which indicates low utility [1]. For more stringent budgets, $\varepsilon = 0.05$ the utility of DPT further degrades to 0.73 JSD error. To measure the utility in maintaining frequent patterns, F1 score, a well-known measure, with range $[0,1]$ is used where larger a F1 score indicates a higher utility (see Section 5.6). DPT leads to rather low F1 scores of 0.32 and 0.08 for $\varepsilon = 0.5$ and $\varepsilon = 0.05$, respectively. In other words, DPT fails to preserve the movement pattern of trajectories when publishing a trajectory dataset under differential privacy. Further important measures are the memory footprint and an algorithm's runtime. For a grid cell of size 100×100 m^2 for instance, DPT needs 13 GB RAM, and its runtime exceeds a day already when generating 100,000 trajectories. This memory usage increases by a factor of 4 when the cell size decreases to 50×50 m^2.

Furthermore, the state-of-the-art mechanisms cannot preserve the semantic properties of trajectories such as stay locations. However, such semantic properties are essential for many applications. For example, the amount of time an individual has stayed in a location identifies the place of interest for that person. Such places of interest reveal vital information for understanding the daily habits of individuals [23] [24]. Preserving the semantic properties of trajectories under differential privacy is a gap in the current literature.

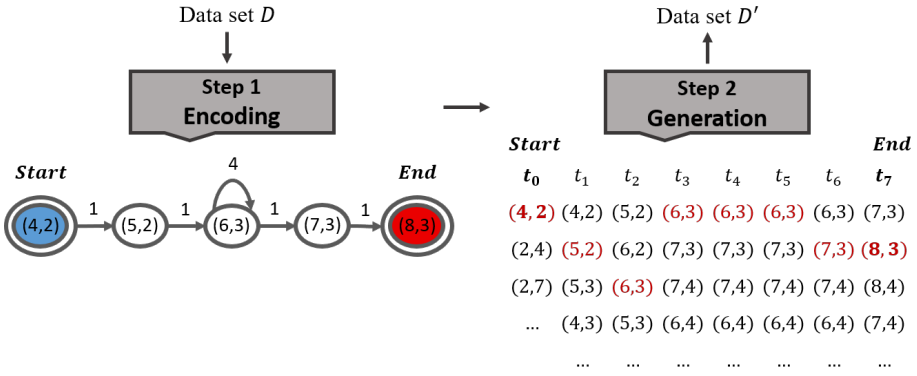
Contributions. In this chapter, we propose a novel 2-stage mechanism for publishing large trajectory datasets under ε-differential privacy. Our key contributions are:

- Effectively encoding trajectories as a graphical generative model. In contrast to the state-of-the-art model, our proposed encoding strategy exploits the trajectory distribution (data-

[1] The original paper [20] uses JSD with value range $[0, ln(2)]$. We instead use JSD with $[0, log_2(2)]$ bound which is common in computing statistical distance between two distributions [80], [81].



(a) Real dataset D (b) Synthetic dataset D'



(c) Mechanism flowchart

Figure 5.1: Overview of TGM system with an example.

adaptive), which leads to high scalability and can accurately capture the properties of trajectories such as travelled distance or movement patterns.

- Introducing a highly efficient algorithm for generating trajectories with ε-differential privacy guarantee that is more than 4000 times faster than the state-of-the-art algorithms while reducing memory requirements by a factor of around to 50.
- In contrast to existing work, our approach can generate trajectories of arbitrary length while maintaining travel distance and frequent trajectory patterns of the original trajectories. To the best of our knowledge, TGM is the first mechanism that also captures the stay locations of trajectories.
- Our extensive experiments show that our mechanism improves the utility of published trajectories typically around an order of magnitude without any further assumptions such as the trajectory length or the velocity of objects.

Algorithm 7 Trajectory Generative Mechanism (TGM)

Input: Trajectory dataset D,
Uniform grid S,
Min-stay time λ,
Privacy budget ε,
Number of synthetic trajectories N,
Output: Synthetic trajectory dataset D'

1: $G = Encoding(D, S, \lambda)$
2: $D' = Generation(G, \varepsilon, N)$
3: **return** D'

Mechanism Overview

We propose *Trajectory Generative Mechanism* (TGM) for publishing trajectories under differential privacy as depicted in Algorithm 7. It shows our two-stage mechanism that takes as input a dataset D of trajectories and outputs a synthetic dataset D' generated through an ε-differentially private process. Each trajectory is a sequence of locations with longitude and latitude values which are modeled using a graphical model in line 1. The other inputs are a uniform grid S and min-stay time λ for modeling the data (see Section 5.3), and privacy budget ε and the number of trajectories N to be generated by the algorithm (see Section 5.4). Figure 5.1 describes an overview of our mechanism with an example. TGM is consists of two stages of encoding (modelling) and generation. The second stage uses ε budget to generate trajectories while satisfying ε-differential privacy. In particular, the noise added to achieve privacy in TGM is adapted to the *process* of generation (second step) instead of the model (first step) used in existing works.

Stage 1: Encoding Trajectories

In the first step, we propose a probabilistic graphical generative model G that accurately encodes the given trajectories. The basic intuition is to provide aggregated statistics of the data to capture any movement behaviour in trajectories. A moving object may have *moved* to new locations at each timestamp or it may have *stayed* in a location for some time intervals. In our novel encoding model (see section 5.3), we formulate the move and stay episodes in a trajectory and use the corresponding statistics to construct the model. Our model fits a trajectory to *Markov process* in which next location can be estimated using recently visited locations in the trajectory.

Stage 2: Private Generation

Given the graphical model G, the second step generates synthetic trajectories. Rather than simply adding noise to the values of the model to achieve ε-differential privacy, we propose a novel algorithm (in section 5.4) that adaptively adds noise to the process of generating a trajectory. Randomizing the process instead of the model values enables our algorithm to maintain the movement pattern of trajectories with arbitrary length. Given i previous locations of the generated trajectory, we compute the transition probabilities at next step: the probability of *moving* to a neighbouring location, *staying* at the same location, or terminating the trajectory at this point. Then, the Exponential mechanism [17] with *adaptive* ε' is used to privately select a location where $\sum \varepsilon' = \varepsilon_1$ is the total budget allocated for selecting successive locations in a trajectory.

However, the Exponential mechanism should be carefully used such that ensure the algorithm can capture long trajectories with high utility. To achieve this, we propose a *directed budget restoring* technique that allows saving the privacy budget when the direction change in successive locations is larger than a threshold. Conceptually, a moving object tends to continue the same direction most of the time, except it needs to turn. Since the threshold reveals information about the correct movement directions, we use Laplace mechanism [22] with ε_2 to add noise. The total budget is split between the Exponential mechanism and the Laplace mechanism such that $\varepsilon_1 + \varepsilon_2 = \varepsilon$ The process of generating a trajectory stops when it reaches the end of trajectory or ε is consumed completely.

The following example describes the two steps of our mechanism.

Example 5.1.1. *Figure 5.1 (a) shows a trajectory dataset D in which a trajectory may have stopped at some regions between source and destination, for example, for shopping, drinking coffee or withdrawal from a bank. A sample stay location is displayed as a balloon showing the trajectory has stayed in the shopping center for a while. Simply coarsening the trajectories by mapping them to a grid cell would destroy this information. Figure 5.1 (c) shows the steps of generating trajectories while ensuring to maintain all relations as well as the regions of interests. TGM first fits the trajectory to a complex network shown in* Step 1*, Encoding. A node represents a grid cell (i.e., a location in the coarsen spatial domain) that a trajectory may have crossed over or stayed there for some times. A directed edge between two adjacent nodes shows the object has moved from the one node to the other. The two concentric ovals depicted in blue and red are virtual nodes*

and maintain the start and end location of all trajectories, respectively. The table shown in Step 2*, Generation, depicts the process of generating a trajectory. Starting from a random start point* $(4,2)$ *at time* t_0*, the generator selects the next location* $(5,2)$ *at time* t_1 *based on the probability of moving to one of its neighbouring locations* $\{(5,2),(5,3),(4,3),(3,3),(3,2),(3,1),(4,1),(5,1)\}$ *or staying in the current location* $(4,2)$*. The probabilities are perturbed by noise to ensure differential privacy. Each trajectory is generated independently by a total* ε *budget. At each iteration of generation process, a portion of the budget is used to perturb the probabilities. The generation process stops when it reaches the end of the trajectory,* $(8,3)$ *at time* t_7 *in this example, or the* ε *is consumed completely. Figure 5.1 (b) shows the generated dataset* D' *with synthetic trajectories that maintain the key properties such as travelled distance and regions of interests.*

The rest of the chapter is organized as follows. Section 5.2 reviews the application of graphical generative models in publishing trajectories. In Section 5.3, we formulate trajectories and introduce our spatial-temporal discretization followed by the encoding step of TGM mechanism. Section 5.4 presents our private generation process and extensive evaluations are discussed in Section 5.6. We conclude the chapter with future directions in Section 5.7.

5.2 Review of Graphical Models on Trajectories

Trajectory datasets provide massive movement data which can easily exceed millions of locations and trajectory segments. The uniqueness of each location in a trajectory makes challenges in modelling and analyzing the trajectories. Due to the uniqueness in locations, the trajectories are not directly comparable and calculating the intersections between segments of all trajectories is computationally expensive.

Instead of grouping the (sub-)trajectories and comparing them based on some characteristics such as time difference or velocity, recent studies think of trajectories as a complex network of connections in time and space with no underlying topology which can be accurately modelled by a graph [82]. Adopting the "graphical perspective" provides flexibility in defining edges for transitions from location A to B in a trajectory. It also models the correlation of route similarity between trajectories, which are crucial for extracting group behaviours and spatial patterns from huge datasets [83].

Considering the spatial regions as nodes in the graph, the transition between adjacent nodes could be represented by directed edges. We can then treat the nodes as transition states in a Markov chain where the edges represent the probability of transitioning between the states. Calculating the movement counts across successive time steps accompanied by the direction of movement enables a graphical model to accurately model the movement patterns through time and identify the *move* or *stay episodes* in trajectories: transition to the same state depicts a stay episode versus transition to a new state shows a move episode in a trajectory.

Since the graphical model accurately captures movement patterns and the flow of information across the spatial-temporal domain, it can be used to model the distribution of trajectories of a large dataset such that synthetic trajectories could be generated accurately. However, generating the trajectories under differential privacy entails three critical challenges: (1) Budget allocation. As mentioned above, existing mechanisms rely on the assumed maximum length of trajectories to allocate a portion of the privacy budget to each step of generation. However, in a graphical model, there is no assumption about the trajectory lengths. Hence, choosing a budget allocation strategy is non-trivial; (2) Spatio-temporal aggregation resolution. The density or sampling interval of data can strongly influence the structure of the graph. It is essential to consider an adaptive strategy for discretizing the spatio-temporal domain with respect to the data type; (3) Generation strategy. Conventional approaches in generating trajectories, add noise to the values in the constructed model and then use the noisy values in the generation process. However, this approach is not practical in a graphical model due to the unbounded trajectory length and dependency of all nodes in the graph that causes adding a large magnitude of noise to the generated trajectories.

5.3 Preliminaries

5.3.1 Trajectory Dataset

Given a spatial domain $\mathscr{S}$ and time domain $\mathscr{R}$, we define a trajectory as a sequence of $(s, t) \in \mathscr{S} \times \mathscr{R}$ in which s represents the location of trajectory at time t with longitude and latitude coordinates. Formally, a trajectory is $T = \{(s_1,t_1),\ldots,(s_n,t_n)\}$ where $\forall_{1\leq i<n}\, s_i \in \mathscr{S}$ is a location (not necessarily distinct) and $\forall_{1\leq i<n}\, t_i \in \mathscr{R}$ denotes a timestamp such that $t_i < t_{i+1}$. n denotes the length of trajectory that is the number of tuples or alternatively, locations. Considering $t_1 = 1$,

next time stamps are relative to t_1, sequentially increasing by one. A trajectory dataset D of size $|D|$ is a set of trajectories with arbitrary lengths. Using the relative time and considering a location as a state of trajectory at a specific time, we see a trajectory T of size n as a sequence of states $T \in \mathscr{S}^n$ where the sate may change randomly from time t_i to t_{i+1}. Given a sub-trajectory $T^{[i-k+1,i]} = \{(s_{i-k+1},t_{i-k+1}),\ldots,(s_i,t_i)\}$, the sequence of last k states in a trajectory at time i, we can predict the next state in trajectory according to $k-$order Markov chain [84]. For simplifying the notation, we use T^i when $i-k+1$ refers to (s_1,t_1), the the first tuple in trajectory.

Definition 5.1. *(Markov Chain) Let $T = \{(s_1,t_1),\ldots,(s_n,t_n)\}$ be a trajectory with finite states taking value in $\mathscr{S}$. The trajectory follows $k-$order Markov chain such that given a sequence $T^{[i-k+1,i]}$ for $k \leq i \leq n$ and $s \in \mathscr{S}$, we have*

$$P_{i+1}(s_{i+1} = s \mid s_{i-k+1},\ldots,s_i),$$

where P_{i+1} is the transition probability *of moving to state s at time t_{i+1} given the last k observations by time t_i.*

The transition probability depends on the movement behaviour of trajectory. A trajectory may have repeated locations in consecutive time stamps meaning the moving object has *stayed* at a specific location. For example, a trajectory $T : \{(s_1,t_1),(s_2,t_2),(s_2,t_3),(s_2,t_4),(s_3,t_5)\}$ denotes the object has *moved* to location s_2 at t_2 and *stayed* at s_2 during $[t_3,t_4]$ time interval. With this observation we define two episodes of movement in a trajectory as follows:

Definition 5.2. *(Trajectory Episodes) Given a spatio-temporal trajectory $T = \{(s_1,t_1),\ldots,(s_n,t_n)\}$, the movement episodes are:*

- Stay episode*, is a sub-trajectory $T_s^{[i,j]} \subseteq T$ with consecutive time stamps but the same location, i.e., $T_s^{[i,j]} = \{(s_i,t_i),\ldots,(s_j,t_j) \mid 1 \leq i < j \leq n, \forall_{i\leq k,k'\leq j}\ s_k = s_{k'}\}$*
- Move episode*, is a sub-trajectory $T_m^{[i,j]} \subseteq T$ with consecutive time stamps but no two equal locations, i.e., $T_m^{[i,j]} = \{(s_i,t_i),\ldots,(s_j,t_j) \mid 1 \leq i < j \leq n, \nexists_{i\leq k,k'\leq j}\ s_k = s_{k'}\}$*

In the above example, $[t_3,t_4]$ is a stay episode with the stay location s_2 while $[t_1,t_2]$ and $[t_5]$ depict two move episodes in T.The stay episode determines the transition probability of staying at last visited state while the move episode identifies the probability of moving to a new state. Note

that we assume there are no loops in trajectories. Let $c(D, T^i)$ denote the number of trajectories in D with sub-trajectory T^i. For a dataset D, we estimate the transition probability of a state $s \in \mathscr{S}$ at time $i+1$ as

$$P_{i+1}(s_{i+1} = s \mid s_{i-k+1}, \ldots, s_i) = \frac{c(D, T^i s)}{c(D, T^i)}.$$

Note that in the indexes used for a trajectory, the temporal component is used only for aggregation and encoding. The generation steps use the ordered locations (spatial component) only. Since the Markov chain concept is used for generation, the temporal component is represented as the subscript of a location in the trajectory showing that we are not using Markov chain concept for estimating temporal component.

Using the movement episodes and the transition probabilities, our goal is first to develop a structure for maintaining the statistics and second propose a generative model for privately constructing trajectories from the data structure. In the following, we describe an aggregation strategy to transform the trajectory dataset into a graphical model and capturing the distribution of a dataset. Later we use the graph to develop our generative model for privately constructing trajectories.

5.3.2 Spatial and Temporal Aggregation

The (s,t) tuples in a trajectory are taken from continuous spatial and temporal domains $\mathscr{S} \times \mathscr{R}$ causing the trajectories not to be directly comparable. To analyze trajectories, it is necessary to aggregate the spatial domain into regions and the temporal domain into intervals [82]. The aggregation, however, is non-trivial and depends on the data context. For instance, simply discretizing the spatial domain with random size regions may not effectively capture movement patterns in trajectories. Note that spatial aggregation would lead to coarsening of the granularity of the temporal domain. Eliminating tuples locating in a region and representing them by only the region's centroid, coarsens the granularity of the temporal domain as well. The temporal aggregation should be done carefully; otherwise, it may eliminate tuples representing the stay episode in the trajectory. In this chapter, we introduce a novel aggregation strategy which is *data-adaptive*, based on public information and incorporate both spatial and temporal elements of data. We first discuss the spatial aggregation and then define the temporal aggregation definition.

In spatial aggregation, we define *closeness* between locations and aggregate the close locations

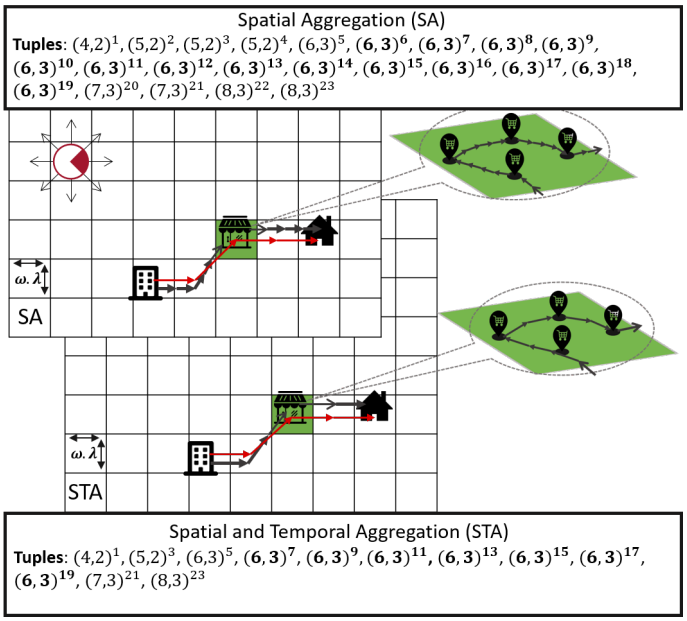


Figure 5.2: An example of spatial and temporal aggregation.

as a *region*. Given a region, its centroid represents all locations in the region.

Definition 5.3. *(Aggregated Spatial Domain) Let $\mathscr{S}$ be the continuous spatial domain. The aggregated (discretized) spatial domain is $S \subset \mathscr{S}$, the set of regions' centroids.*

Defining the closeness, however, depends on the data context. We incorporate the temporal element of trajectories into spatial aggregation to effectively formulate a data-driven size for the regions with no privacy cost. For example, given taxicabs trajectories with average velocity of $v = 36\ km/h$ $(600\ m/min)$, one may set regions to 1 km^2 areas to capture *significant movement* patterns per hour. However, this measure may not maintain information about stay locations. If the taxi has stayed in a location for 2 minutes (assuming data is sampled every 2 minute), its trajectory has 2 locations inside the region. Space aggregation only based on velocity cannot maintain all the locations as they are represented by the region's centroid. We need to define a measure that takes the stay episodes into account while calculating the regions' size.

Given a uniform sampling rate of data, we introduce a *Min-Stay Time parameter* λ depicting the minimum time period that a moving object has to be in a location to be considered as a stationary object. The value of this parameter should be determined based on data content. In the above example considering $\lambda = 5$ minutes means that if there are at least two tuples (s_i,t_i) and (s_j,t_j) in the trajectory such that $t_j - t_i = 5$ minutes and $\forall_{i \leq k < k' \leq j} s_k = s_{k'}$, the taxi has stayed in location l_i.

Definition 5.4. *(Min-Stay Time) Let $T = \{(s_1,t_1),\ldots,(s_n,t_n)\}$ be a spatio-temporal trajectory. If there is a sub-trajectory $T^{[i,j]}$ with $t_j - t_i \geq \lambda$ and $\forall_{i\leq k<k'\leq j} s_k = s_{k'}$, $T^{[i,j]}$ represents a stay episode in T with the min-stay time λ.*

Given λ, the size of a stay episode in a sub-trajectory can be calculated as $|T_s^{[i,j]}| = (t_j - t_j)/\lambda$.

Using the velocity and λ as two temporal features, now we can calculate an effective size for regions. Intuitively, the region should be adequately large to cover the distance that the object would traverse without staying in any location of that region. If a taxi, for instance, is driving with $\omega = 600m/min$ and $\lambda = 5$ minutes, considering the regions with area larger than $(600\times 5)^2\ m^2$ ensures that no *significant movement* is missed and at the same time it can cover the stay locations if any has happened.

With this intuition, we map a uniform grid on the spatial domain in which a region's area is set to

$$(\omega \times \lambda)^2,$$

and each region is represented by its centroid. Note that the velocity of an object may change regarding the speed limits in different regions resulting in regions of different sizes. Besides, the shape of regions should not be necessarily square. Without loss of generality, we estimate the movements with the *average speed* of the objects and a square region of fixed size uniformly throughout the spatial domain. Let $f(\cdot)$ be the function for spatial aggregation. For each trajectory, the function maps a location to its closest region's centroid. If a point has an equal distance to more than one region, a consistent strategy is used to assign it to a region. The gaps are filled consistently for jumps over regions due to speeds higher than the average value.

By this mapping, we limit the transitions from a cell to only its 8 adjacent cells in the grid. A moving object at cell a can stay in the cell at the next timestamp or move to one of the 8 *neighboring* cells of a. All the other transition probabilities are considered as 0. Thus, given a grid S consisting of $|S|$ cells, the total number of transition probabilities for a Markov model of order k would be $O(9^k|S|)$. This number is significantly less than $O(|S|^{k+1})$ where a moving object can transit to $|S|$ cells at each timestamp.

Although the spatial aggregation ensures preserving the stay episodes, the tuples within a stay episode do not provide significant information about the movement of objects. Assume a stay episode $T_s^{[i,j]}$ in a region with time interval $\lambda = j - i$ meaning the stay episode is taken exactly

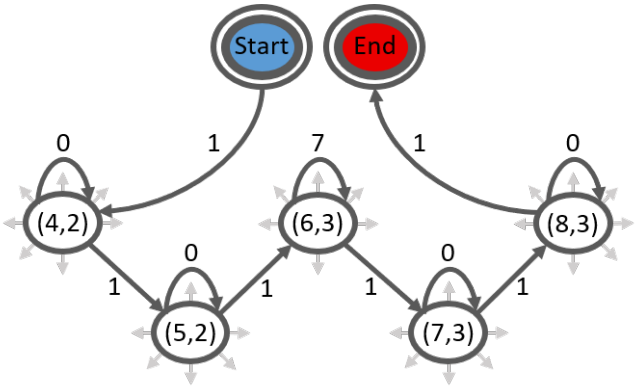
one λ from tuple (s_i,t_i) to tuple (s_j,t_j). When the sampling rate of data is faster than λ (sampling rate $< \lambda$), there are some tuples $(s_k,t_k) \in T_s^{[i,j]}$ where $i < k < j$ and the tuple do not provide extra information about the stay episode. The same concept applies for a move episode $T_m^{[i,j]}$. This observation motivated us to define our *temporal aggregation* strategy with the goal of eliminating *insignificant tuples* from trajectories in the dataset.

Definition 5.5. *(Temporal Aggregation) Let D be a trajectory dataset with the min-stay time λ. The temporally aggregated dataset is denoted by $D^* \subset D$ such that $\forall_{T \in D, T^* \in D^*}\ T^* \subseteq T$ and $\forall_{i,j \in T^*} |t_i - t_j| = a.\lambda$ for $a \in \mathbb{N}$.*

Temporal aggregation applies on each trajectory individually and is agnostic to potential intersections between trajectories. Briefly, given a min-stay time, consecutive locations in a trajectory which their temporal component is in a particular time interval, regarding the min-stay time, are aggregated and considered as a single location in the trajectory.

Let $f(\cdot)$ and $g(\cdot)$ be the spatial and temporal aggregation functions, respectively. The goal is to transform a trajectory $T \in D$ to the aggregated domain such that for each tuple $(s_i,t_i) \in T$: (1) the location s_i is replaced by the closest region's centroid; (2) if the time component t_i is located inside a λ time interval, i.e., $|t_i - t_0| \neq a.\lambda$ for $a \in \mathbb{N}$, the tuple is eliminated from the trajectory. The aggregated trajectory is a sequence of segmented tuples where a segment represents a stay episode when the contiguous tuples have the same location, and represents a move episode, otherwise. We illustrate our aggregation strategy in the following example.

Example 5.3.1. *Figure 5.2 shows a sample trajectory aggregated first on the spatial component and next on the temporal component. The uniform grid labeled as "SA" depicts a spatial aggregation where the region size is $(velocity\,.\,min\text{-}stay\ time)^2 = (\omega.\lambda)^2$. Assume the trajectory represents the movement of a vehicle that starts from office, drives to a shopping center and ends at the home location. Arrows show the movement direction of the vehicle. The length of an arrow represents the temporal distance of two tuples in a trajectory where shorter means smaller distance and vice versa. Spatially aggregated tuples are shown in the form of $(x,y)^{time}$ in which x and y are the horizontal and vertical indices of the corresponding region in the grid. The red arrows represent the mapping of a location point to a region's centroid after spatial aggregation. The green region is zoomed in to emphasis on multiple tuples locating inside the region, i.e., shopping center. The*



(a)

v_{id} = Start	
v_{id}	count
(4,2)	1
(4,4)	1
(2,7)	1
...	...

v_{id} = End		
v_{id}	prefix	count
(8,3)	(4,2), (5,2), (6,3), (7,3),(8,3)	1
...	...	...

(b)

$v_{id} = (4,2)$		
prefix	count	stay time
(4,2)	1	0
...	...	...

$v_{id} = (6,3)$		
prefix	count	stay time
(4,2), (5,2), (6,3)	1	7
...	...	...

$v_{id} = (8,3)$		
prefix	count	stay time
(4,2), (5,2), (6,3), (7,3), (8,3)	1	0
...	...	...

(c)

Figure 5.3: Our graphical generative model and the information stored in the nodes. (a) is a branch of graph representing the data of a single trajectory. (b) and (c) show the tables of data stored in v_{start}, v_{end} and some other nodes of graph. v_{id} in a table depicts the node identifier.

tuples corresponding to the stay episode inside this green region are shown in bold. Each tuple represents an arrow in the trajectory. The two arrows depicting the move episodes of entering the green region and then exiting it are shown in a different style (on the grid as well as the zoomed balloon). The grid labeled as "STA" shows the generated trajectory after spatial and temporal aggregation. Assuming the min-stay time $\lambda = 2$ minutes and the scale of timestamps in a minute, in the temporal aggregation the tuples with an even time stamp are eliminated. The reason is that by starting from the first timestamp in trajectory, these tuples are placed within a 2 minutes time interval. Longer arrows in "STA" comparing to "ST" depict the 2 minutes time interval between contiguous tuples.

5.3.3 Graphical Generative Model

Given the aggregation strategy, we propose our *graphical generative model* that carefully captures the distribution of trajectories on the aggregated space. We model trajectories as a complex network with a non-trivial topology on the spatial and temporal space. Despite the conventional hierarchical models such as prefix trees that simplify or make assumptions about the movement behaviour, a complex network can precisely describe any movement pattern in trajectories and their interconnections. The complex network is represented by a directed graph $G = (V, E)$ where V and E are the set of nodes and edges, respectively. We first describe the graph structure and then formulate transition probabilities based on the graph to generate trajectories.

Let $T^i = \{(s_1, 1), \ldots, (s_i, i)\}$ be the prefix of aggregated trajectory T by time i in which $s_k \in S$ is a region that the trajectory has passed through at time k. In our graph $G = (V, E)$, each node $v \in V$ represents a region $s \in S$ in the spatial grid, i.e., $|V| \propto |S|$. Each node v maintains two key information about a passing trajectory T of size n: (1) the prefix $T^i \subseteq T$ of the trajectory *entering* the node v at time i and (2) the *stay time* of trajectory at node v, i.e., $|T_s^{[i+1,j]}|$ for the stay episode $T_s^{[i+1,j]} \subseteq T$ where $i < j \leq n$. Note that at time i trajectory enters the node, and hence is in move episode. The stay episode can start at $i+1$, after entering the node. The first component, maintains the *move episode* statistics that may span multiple nodes in a graph while the second component records the statistics of *stay episode* per node.

Our aggregation strategy (see Section 5.3.2) transforms the dataset into a discrete spatial and temporal domain. By this transformation, the degree of each node $v \in V$ is bounded proportional to the number of *neighboring* nodes of v in graph G. In particular, a trajectory visiting node v at time i, has two options for expansion at time $i+1$: (1) staying at the same node v (stay episode), or (2) moving to a *neighbor* node $v' \in V$ (move episode). Referring to the spatial aggregation grid in Figure 5.2, each node v has 8 neighbors. Hence, each node $v \in V$ has outdegree of size 9, i.e., $deg^-(v) = 9$ including one for self-transition (i.e., staying at same node) and 8 for transition to neighbor nodes. The same concept applies to the indegree of node such that $deg^+(v) = 9$. For brevity we focus on outward edges. Let $e = (v, v') \in E$ be an edge that directly connects the nodes v and v' in graph G. The edge e depicts a self-transition when $v = v'$ and it shows a transition to a neighbor node when $v \neq v'$. Given the bounded degree, the information maintained in nodes, and the inherited sequentiality between the nodes, the graph structure satisfies a Markov chain

described in Section 5.3.1. We can then use a k-order Markov chain to estimate a trajectory's state at each time step. The value k may increase by growing the trajectory over time, and the estimation process works for trajectories of various lengths. The transition probabilities can then be computed using *graph count queries* of form $\mathscr{G}(m,v,T^i)$ where $T^i \subseteq T$ is a prefix of size k and v is a graph node. The query counts the number of trajectories with prefix T^i at a specific node v, i.e., $c(T^i,v)$. The count represents the statistics of a move episode at time i if $m=1$ and a stay episode if $m=0$.

Definition 5.6. *(Graph Count Query $\mathscr{G}(\cdot)$) Let $G=(V,E)$ be a graphical model constructed based on a trajectory dataset where $v \in V$ in a node in G. A graph count query $\mathscr{G}(m,v,T^i)$ is defined as*

$$\mathscr{G}(m,v,T^i) = \{c(v,T^i) \mid T^i = \{(v_1,1),\ldots,(v_i,i)\} \models \begin{cases} v_i \neq v_{i-1} & m=1 \quad \# \textit{move episode} \\ v_i = v_{i-1} & m=0 \quad \# \textit{stay episode} \end{cases}, v \in V\}.$$

We add two nodes v_{start} and v_{end} to our graphical model to maintain the statistics of starting and ending locations of trajectories. The start node $v_{start} \in V$ records the count of trajectories starting at each region $s \in S$. The end node $v_{end} \in V$ records the prefix of trajectories ended at a region $s \in S$ with their corresponding counts.

In the following example, we illustrate our graph structure with a sample trajectory. This section is concluded with our algorithm for constructing the graphical generative model on the aggregated space. The next section describes reconstructing trajectories using the model and how we ensure ε-differential privacy throughout the reconstruction process.

Example 5.3.2. *In Figure 5.3(a) the sample trajectory transformed in Example 5.3.1 is recorded as a sequence of nodes in the graphical model. Each node is specified by an identifier v_{id}. Light grey arrows show the node is also connected to other neighbors who are not displayed for brevity. Due to the aggregation, each node has 8 neighbors. v_{start} and v_{end} are special nodes depicted by two concentric ovals in blue and red, respectively. v_{start} has a direct link to all inner nodes of graph (i.e., $V' = V - \{v_{start}, v_{end}\}$). The corresponding table in Figure 5.3(b) shows a part of*

stored data in v_{start} based on our sample dataset. The v_{end} table stores the information of nodes that trajectories are terminated there. v_{end} also records the prefix of these trajectories as well as the number of trajectories with that prefix. The corresponding tables of inner nodes are specified by the index of regions in the spatial domain. Some of the tables are shown in Figure 5.3(c). Each table records the trajectory's prefix *by current node, the* count *of trajectories with this prefix passing the node, and the total number of timestamps that a trajectory have stayed in the node as* stay time. *The numbers on edges in Figure 5.3(a) show the number of trajectories transitioned from a node to a neighbor node. Self-transition edges represent the state in which a trajectory has stayed in a node, and the corresponding count shows the stay time of that trajectory at that node. In the sample trajectory in Example 5.3.1, the stay time at node $v_{id} = (6,3)$ is 7 which is shown on the corresponding self-transition edge of this node.*

The graph structure enables us to (1) model the sequentiality and distribution of trajectories with arbitrary lengths; (2) capture interconnection patterns between trajectories; and (3) distinguish stay episodes versus move episodes. Algorithm 8 describes building our graphical generative model based on the aggregated space. The inputs are the trajectory dataset D, min-stay time λ and the uniform grid S with regions of size $\omega.\lambda$. For each location loc in a trajectory T, the temporal aggregation function $g(\cdot)$ checks whether the location should be eliminated with respect to the prefix of the trajectory by this point. If the location should not be removed, we map the location to the closest region using spatial aggregation function $f(\cdot)$. The index of the region is used as its identifier id to make a new node for the graphical model. If it is the first time, the graphical model is visiting this region, a new node is added to the graph and the count of corresponding prefix increments by 1. Otherwise, we check if the trajectory has recently visited the region. In particular, if the last node in the prefix is the same as the new node, it means the trajectory has stayed in the region and the algorithm increments the stay time corresponding to the prefix. Otherwise, it is the first time that the trajectory has moved to this region. Hence, the algorithm adds this region to the prefix and increases the count of trajectories with this prefix visited this region.

Given the graphical model and the graph count query, we next describe our private generation process that constructs trajectories using the graphical generative model while ensuring ε-differential privacy.

Algorithm 8 Encoding: Graphical Generative Model

Input: Trajectory dataset D,
Uniform grid S,
Min-stay time λ
Output: Graphical model $G = (V, E)$

for trajectory T in D **do**
  $prfx = []$
  for location loc in T **do**
    if $g(loc, prfx, \lambda)$ **then** # Temporal aggregation
      $continue$
    end if
    $id = f(loc, S)$ # Spatial aggregation
    $newPrfx = append(prfx, id)$
    $node = Node(id)$
    if $IsNew(node, G)$ **then** # First visit of node
      $Add(node, G)$
      $IncCount(newPrfx, node, G)$
    else
      if $lastVisit(id, prfx)$ **then** # Stay episode
        $IncStime(prfx, node, G)$
      else # Move episode
        $prfx = newPrfx$
        $IncCount(newPrfx, node, G)$
      end if
    end if
  end for
end for
return $G = (V, E)$

5.4 Private Model Learning

As mentioned above, three main components should be learned in a trajectory generation process: (1) utility of initial locations; (2) transition probabilities; and (3) termination condition. These generation components are learned from the original trajectory data by estimating the counts in the graphical model. To ensure using the statistics does not raise any privacy concern, we employ ε-differential privacy in the *generation process*. Constructing a differentially private graphical model is considerably expensive due to the dependency between the graph nodes. In particular, trajectories are of arbitrary length and have arbitrary interconnections that result in the large sensitivity of graph count query (see Definition 5.6).

Algorithm 9 shows our novel generation process that: (1) ensures end-to-end privacy from

the initial step to the termination of trajectory; and (2) dynamically adjusts the privacy budget of each step that enables generating trajectories of various lengths. As the synthetic trajectories are generated independently, we only need to ensure the generation process of each trajectory satisfies ε-differential privacy. The algorithm takes the graphical model $G = (V, T)$, total budget ε and the number of synthetic trajectories N as inputs. The output is D' — the dataset of synthetic trajectories. The generation of each trajectory T starts by identifying the first location and proceeds with selecting successive locations. $First(\cdot)$ computes a vector of potential first locations with their corresponding utility. Similarly, $Next(\cdot)$ takes the generated trajectory of T by this stage and generates a vector of potential next locations and their corresponding utility. To ensure privacy, the Exponential Mechanism [17] is used for randomly selecting one of the potential locations. Depending on the recent movement pattern, the selected location by Exponential Mechanism might be withdrawn, and instead, $EstLoc(\cdot)$ function estimates the next location. Intuitively, when the object is moving in a fixed direction, we can restore the privacy budget by estimating the next location. The function $DirChng(\cdot)$ computes the direction change. If this change is *less* than a deviation threshold θ it implies the movement direction has not significantly changed (see Section 5.4.3), and the output of $EstLoc(\cdot)$ is used as the next location and the privacy budget used at the step restores to be used later in next steps. Otherwise, the selection location by Exponential mechanism is used. To ensure privacy, the deviation threshold is perturbed by a random value driven from Laplace distribution. To provide ε-differential privacy guarantee, we split the total budget into $\varepsilon_1, \varepsilon_2$ with ratio r such that $\varepsilon_1 = r.\varepsilon$ and $\varepsilon_2 = \varepsilon - \varepsilon_1$. The portion ε_1 is used for selecting locations in each iteration by the Exponential Mechanism and ε_2 is used for adding noise to the deviation threshold using the Laplace mechanism. In $Rtn(\cdot)$, we introduce an adaptive budget allocation approach that determines a portion of ε_1 for selecting the next location.

In the following, we describe the algorithm steps in detail. We summarize the differential privacy definition followed by our novel private generation process. Next, we formulate computing the transition probabilities. Identifying the initial location, selecting the successive locations, measuring the movement pattern and changes in direction are later described in this section.

Algorithm 9 Generation: Private Process of Construction

Input: Graphical model $G = (V, E)$,
Privacy budget ε,
Number of synthetic trajectories N
Output: Synthetic trajectories D'

Let $V' = V - \{v_{start}, v_{end}\}$.
Let ε_1 be for selecting locations and ε_2 be for direction changes detection such that $\varepsilon_1 + \varepsilon_2 = \varepsilon$.
for trajectory T in $[T_1, \ldots, T_N]$ **do**
  $\varepsilon_0 = \varepsilon_1 / 3$
  $\mathbf{f} = First(v_{start})$
  $loc = EM(\mathbf{f}, \varepsilon_0)$
  $Add(T, loc)$ # First location
  $i = 1$
  $\varepsilon' = Rtn(2\varepsilon_1/3, i)$
  while $\varepsilon' > \eta$ *and* $loc \neq v_{end}$ **do**
    $\mathbf{p} = Next(T, V', v_{end})$
    $newloc = EM(\mathbf{p}, \varepsilon')$
    if $DirChng(T, loc, newloc) < \theta + Lap(\varepsilon_2)$ **then**
      $newloc = EstLoc(T, loc)$
    end if
    $Add(T, newloc)$
    $loc = newloc$
    $i = i + 1$
    $\varepsilon' = Rtn(2\varepsilon_1/3, i)$
  end while
  $Add(D', T)$
end for
return D'

5.4.1 Differential Privacy Settings

Let D and D' be two trajectory datasets differing in only one trajectory such that $|(D - D') \cup (D' - D)| = 1$. In this chapter, we consistently consider one trajectory per person. According to ε-differential privacy, the distribution of a randomized mechanism $\mathscr{K}$ on D and D' is not *significantly* different and is bounded by ε.

A mechanism q can be a graph count query (see Definition 5.6). Then, the sensitivity of q, denoted by Δq, is the maximum changes on the query answer by adding or removing a trajectory which depends on the trajectory lengths in a given dataset. Exponential Mechanism and Laplace Mechanism are used in this chapter to achieve ε-differential privacy. The global sensitivity Δu Exponential Mechanism is defined over the neighborhood of trajectory dataset D. We normalize

trajectories such that adding or removing a trajectory changes the counts in the graphical model by 1. With this normalization we ensure $\Delta q \in [0,1]$ and in turn $\Delta u \in [0,1]$. We use $Lap(\sigma)$ to denote the Laplace probability distribution with mean 0 and scale σ. See Chapter 2 for definition of differential privacy and the well-known mechanisms to achieve it.

5.4.2 Learning Transitions

Transition probabilities. Given the generative graphical model $G = (V,E)$, we aim to construct a synthetic dataset of trajectories. The generation process is independent for each trajectory and comprises three steps: (1) selecting a starting location, (2) continuously selecting successive locations which form the trajectory route, and (3) choosing the end location where the trajectory terminates. Let $V' = V - \{v_{start}, v_{end}\}$ be the inner nodes in graphical model. There are $|V'|$ nodes that a trajectory could start. We define the *utility* of a node to be selected as a starting location proportional to its corresponding count stored in v_{start}. Conceptually, a node (or location) where more trajectories are started from has a higher probability to be selected as the starting point of the synthetic trajectory. The *utility* of node $v \in V'$ as a starting point, denoted by $u(v)$, is computed as

$$u(v) \sim \frac{c(v_{start}, v)}{\sum_{v \in V'} c(v_{start}, v)}, \tag{5.1}$$

where $c(v_{start}, v)$ is the corresponding count of node v recorded in v_{start}. Suppose $T^i = v_1, \ldots, v_i$ is the prefix of synthetic trajectory generated thus far. In next time stamp, the trajectory may stay in v_i, move to one of 8 neighbors, or terminate by moving to v_{end}. The probability of staying at v_i such that $v_{i+1} = v_i$ is

$$P(v_i \mid T^i) = \frac{\mathscr{G}(0, v_i, T^i)}{\mathscr{G}(0, v_i, T^i) + \mathscr{G}(1, v_i, T^i) + c(v_{end}, v_i, T^i)}. \tag{5.2}$$

The numerator is the *stay time* of T^i in node v_i. Denominator gets the total count of trajectories with prefix T^i that *passing through* v_i, or *staying at* v_i or terminating there. Assume $a \in nbr(v_i)$ is a neighboring node of v_i. Accordingly, the probability of moving to node a at $i+1$ is

$$P(a \mid T^i) = \frac{\mathscr{G}(1, a, T^i a)}{\mathscr{G}(0, v_i, T^i) + \mathscr{G}(1, v_i, T^i) + c(v_{end}, v_i, T^i)}, \tag{5.3}$$

where $\mathscr{G}(1,a,T^i a)$ gets the number of trajectories with prefix T^i moving from v_i to a. Meanwhile, since the graphical model does not limit the length of trajectories, the termination of trajectory may occur at any time stamp. Thus, given the inner nodes (V') and the statistics recorded in the node v_{end}, the probability of termination at node v_i is computed by

$$P(v_{end} \mid T^i) = \frac{c(v_{end}, v_i, T^i)}{\mathscr{G}(0, v_i, T^i) + \mathscr{G}(1, v_i, T^i) + c(v_{end}, v_i, T^i)}. \tag{5.4}$$

After choosing a start location, the next location v_{i+1} is selected from the potential moves: current location v_i, neighboring locations $nbr(v_i)$, and v_{end}. Selecting v_{end} terminates the generation process. Selection of the current or a neighboring locations means the trajectory is in stay or move episode and the generation process continues to the next time stamp.

Randomized generation. Our generation process is a sequence of differentially private steps with an arbitrary length that satisfies ε-differential privacy. Simply considering a fixed length, for instance, the average length of trajectories in data, is not able to capture various lengths of trajectories in the data distribution. Rather than resorting to assumptions about the length, we choose to incorporate the probability of terminating a synthetic trajectory in the selection process. Therefore, the generation process dynamically determines the length of the synthetic trajectories. This dynamic length determination, however, needs a careful budget allocation at each step. At the beginning, the generation process does not have any information about the target trajectory. As a longer prefix of trajectory is constructed, the generation process gains more information and hence more accurate decision can be made. Thus, adaptive budget allocation is needed to minimize the error due to the randomization noise at each step.

Recall that the generation process starts with selecting the first location and continues by selecting inner locations sequentially until reaching the end node. Let $V' = V - \{v_{start}, v_{end}\}$ be the set of inner nodes in the graphical model $G = \{V, E\}$. For the first location, the utility of each node $v \in V'$ is computed according to Equation 5.1 and then, Exponential mechanism is used to select a location randomly. For the next locations, the utility of all movement options are computed proportionally to Equations 5.2-5.4 and then the Exponential mechanism identifies the next state of trajectory. Considering v_{end} as one of the possible movements at each step enables our generation process to construct synthetic trajectories of various lengths.

Adaptive budget allocation. Let n be the length of the generated trajectory. Total privacy

budget ε should be split between the n steps of generation to ensure ε-differential privacy. If n is known, we could compute a splitting strategy to minimize the error of the generation process. However, n is not known as the process can be terminated at any step. A naive allocation strategy could be considering a fixed ratio of the remaining budget for each step. Such a *fixed* budget allocation fails to address the noise sensitivity of the steps mentioned above. We propose a strategy that adaptively allocates a budget to a step with respect to its sensitivity such that more budget is allocated to the steps that are more sensitive to noise in the generation process.

Given the total length n and a prefix T^i at i^{th} step, next location s_{i+1} is chosen based on the information gained from T^i. Note that trajectories are known to be sparse data, especially in beginning and end locations. The start location should be selected from all locations in the spatial domain, where many counts are small. If because of randomness a location with a count of zero is chosen as the start location, it may significantly degrade the accuracy of the generation process. Thus, selecting the first location is of critical importance and has high sensitivity comparing to the next locations. By growing the constructed prefix, more information is gained about the synthetic trajectory and hence selecting s_{i+1} is less sensitive than s_i.

Let ε' be the allocated budget for selecting i^{th} location such that $\sum_i \varepsilon' = \varepsilon$. With the above intuition, in our proposed budget allocation strategy, we consider the largest budget for the first location and decrease the allocated budget sequentially in next locations such that $\varepsilon'_{i+1} < \varepsilon'_i$. To achieve this strategy we make use of *reciprocals of triangular numbers* with the following series which elegantly provides such property:

$$\frac{1}{3}, \frac{1}{6}, \frac{1}{10}, \frac{1}{15}, \cdots, \frac{2}{(i+1)(i+2)},$$

where the series is convergent to 1. Given this series, we set the budget of first location as $\varepsilon_0 = \frac{\varepsilon}{3}$. The budget of i^{th} step in the generation process is then computed as

$$\varepsilon'_i = \frac{2}{(i+1)(i+2)} \cdot \frac{2\varepsilon}{3},$$

where $\frac{2}{(i+1)(i+2)}$ is the i^{th} sentence in the series. Since the series is convergent to 1, our allocation strategy ensures ε-differential privacy as $\varepsilon_0 + \sum_{i=1}^{n-1} \varepsilon'_i = \varepsilon$. We consider a threshold $\eta = 10^{-4}$ to terminate the generation process when $\varepsilon'_i \leq \eta$. In Algorithm 9, $Rtn(\cdot)$ is the function that computes

the allocated budget for each step of generation process. Note that the process termination happens either because of running out of the budget or selecting v_{end}.

While η applies an upper bound on the length of generated trajectory we introduce an elegant budget restoring technique in the following that enables our algorithm to generate trajectories longer than the upper bound.

5.4.3 Directed Budget Restoring

Suppose the moving object is a vehicle driving in a high-way. The direction of its movement trajectory may not *significantly* change for consecutive long locations. Intuitively, when the direction of trajectory at consecutive locations s_i to s_j where $i < j$ is not *significantly* different, it does not provide new information at interval $(i, j]$. Given the direction at time i, the next locations in the interval $(i, j]$ could be estimated without any new information from the data and hence, spending the privacy budget. Spending privacy budget only for *significant* information gain (called Sparse Vector Technique [85]) is commonly used in dynamic settings when a new observation is added at each timestamp. We employ this concept in our generation process to save the privacy budget only for *significant* changes in the direction of the synthetic trajectory. This would postpone declining the budget towards the threshold η mentioned above and leads to generating longer trajectories. We first encode locations into directions and then, describe our directed budget restoring.

Consider region $(2,5)$ in Figure 5.2 as a node in the graphical model. We encode the node and its neighboring nodes into 9 directions: self-transition as direction $dir = 0$, North-West (node $(1,6)$) as direction $dir = 1$, North (node $(2,6)$) as direction $dir = 2$, etc. This encoding maintains the direction of trajectory at each location in 9 possible transition states. Given the direction dir_i at location s_i and dir_{i+1} at location s_{i+1}, a *significant* change occurs when

$$\Delta dir = dir_{i+1} - dir_i < \theta,$$

where θ is the deviation threshold. Instead of the direction number, we take the *deviation degree* (Δdir°) between the previous step and the new move and use the cosine of Δdir° to identify a significant change in the movement direction. In the above example, let $s_i = (2,5)$ with $dir_i = 1$ and $s_{i+1} = (2,6)$ with $dir_{i+1} = 2$. The object's direction has then changed from North-West to

North. Hence, the deviation degree is $\Delta dir^\circ = 45^\circ$ with $\cos(\Delta dir^\circ) = \cos(45^\circ) \approx 0.707$.

Given the deviation degree at each movement step, we set the deviation threshold as $\theta = 0.707$ in our algorithm. Intuitively, if the deviation degree in the new step is larger than 45°, i.e.,

$$\cos(\Delta dir^\circ) < \theta,$$

we consider the direction of object has significantly changed, otherwise, it is still moving in the previous direction. In our example, the deviation degree does not imply a significant change as $\cos(45^\circ) \approx 0.707 \not< \theta$. In such case, we withdraw the selected location (s_{i+1}) and instead, estimate the location using the previous movement direction at s_i. In our example, the previous direction is $dir = 1$ (i.e., North-West) and hence, we estimate the next location as $s_{i+1} = (1,6)$. As another example, let $s_{i+1} = (3,6)$ is selected. In this case, the new direction is $dir = 3$ (i.e., North-East) with $\Delta dir^\circ = 90^\circ$ and $\cos(90^\circ) = 0 < \theta$ which indicates a significant change in direction. In this case, we use the selected location with the cost of privacy budget. Considering $\theta = \cos(0^\circ)$ corresponds to the basic generation method without considering our directed budget restoring strategy. The function $DirChng(\cdot)$ in Algorithm 9 computes the cosine of deviation degree and $EstLoc(\cdot)$ does the location estimation if the direction change is not significant. We show in our experiments that directed budget restoring can significantly improve the utility of synthetic trajectory.

5.4.4 Direction Weighting in Generation

The generation process is based on the information gained from the constructed prefix T^i by step i. However, the randomness applied to ensure privacy may damage this information, especially in longer prefixes. A common approach to retrieve the directionality of data is to incorporate the direction of visited locations in estimating the next location [20]. We use a similar idea with notable additions. Consider $(2,5)$ in Figure 5.2 as a node in graphical model. We encode the node and its neighboring nodes into 5 directions: self transition as direction $dir = 0$, aggregation of North-West, North and West directions (nodes $(1,6)$, $(2,6)$ and $(3,6)$) as direction $dir = 1$, aggregation of North-East, East and North directions (nodes $(3,6)$, $(3,5)$ and $(3,4)$) as direction $dir = 2$, etc. This encoding maintains the aggregated direction of the trajectory. To prevent sudden unrealistic changes of direction, we modify our generation process to remember the recent

movement directions. Initially, all the directions have equal weight at the first location of synthetic trajectory. Given the prefix T^i of trajectory generated thus far, we set the weight of a direction dir as $w_{dir}(T^i) = \alpha^{c(dir,T^i)}$. $\alpha > 1$ is the weighting factor and $c(dir,T^i)$ represents the number of times T^i has followed direction dir. In order to investigate recent behaviour of trajectory, we use only the window of last m moves of T^i, denoted by $T^{(i-m,i]}$, to compute the weight of direction dir, i.e., $w_{dir}(T^i) = \alpha^{c(dir,T^{(i-m,i]})}$. When $m = 0$ it corresponds to the basic generation process without considering the direction of recently visited locations. We consider $\alpha = 1.4$ and show in our experiments that using weighting transition directions in computing the utility of potential moves in $Next(\cdot)$ can improve the utility of generated trajectories by Algorithm 9.

5.5 Formal Guarantees

In this section, we analyze our mechanism in Algorithm 7 along two key dimensions; computational efficiency and privacy.

5.5.1 Computational Efficiency

The running time of our mechanism described in Algorithm 7 is $O(|D|n + |D'|n)$ where n is a constant representing the maximum length of trajectories. The algorithm is significantly parallel in each step: Adding trajectories to the graphical model in Algorithm 8 and also generating synthetic trajectories in Algorithm 9 can be conducted independently. The encoding step maps each trajectory into a sequence of grid cells in which there is a negligible chance that a trajectory intersects all grid cells especially in low granularities when the number of grid cells is large. Hence, the total size of grid is not a parameter affecting the complexity algorithm. The only required serialization in each step is that the n locations of trajectory must be processed in sequence. We utilize this important feature in the experiments to show the efficiency of TGM. Note that n denotes the *worst case* scenario when all trajectories in the given dataset have the same length equal to the longest trajectory. In practice, the average length of trajectories (in raw dataset and hence, the synthetic dataset) are usually much smaller than n. A remarkable point about our mechanism is that the computational complexity is independent of the problem space. Regardless of the size of spatial space that the dataset is representing, TGM efficiently captures the regions maintaining the data

value ignoring the rest of regions in space. In our experiments, we show that our mechanism works efficiently on large datasets.

5.5.2 Privacy

We show that Algorithm 7 provides ε-differential privacy guarantee.

Theorem 5.1. *TGM satisfies ε-differential privacy.*

Proof. Our mechanism is consist of two main steps: the encoding step constructs the graphical generative model G, and the generation step makes use of G to constructs synthetic trajectories. Since the graphical model is not published, we only need to ensure the generation step in Algorithm 7 (see Algorithm 9 for details) satisfies ε-differential privacy.

According to the composition theories for differential privacy, when using a single dataset, ε values accumulate additively in consecutive queries. In Algorithm 9, we make n calls to the Exponential Mechanism with parameter ε' such that $\sum_{i=0}^{n-1} \varepsilon' = \varepsilon_1$. The threshold θ is perturbed once and using the Laplace Mechanism with parameter ε_2. The noisy threshold is used afterward in the generation process. Recall the normalization of trajectories to 1 bounds the sensitivity of the Exponential Mechanism to 1 as adding or removing a trajectory changes the answer of a graph count query at most by one. The sensitivity of Laplace Mechanism is also bounded to 1 as the direction change is in $[0°, 360°]$ where the cosine value is in $[0, 1]$. Adding the total privacy cost of Exponential Mechanism and Laplace Mechanism, TGM satisfies $\varepsilon_1 + \varepsilon_2 = \varepsilon-$differential privacy. □

5.6 Evaluation

We now evaluate the performance of TGM over two large trajectory datasets. The evaluation results are presented in two parts:

- TGM is significantly scalable comparing to the state-of-the-art method with high utility in generating trajectories. We show that the generated trajectories resemble the original data on three utility measures derived from prior works – travelled distance, frequent patterns, and stay locations.

Dataset	$\|D\|$	$l_{max}(km)$	$l_{avg}(km)$	$st_{mean}(s)$	$area(km)^2$	$\omega.\lambda(m)$
Porto-Taxi	1,710,671	20.69	4.15	531	76×52	420
Melb-Car	10×10^6	100.23	30.86	714	76×105	1,320

Table 5.1: Datasets summary.

- Individual components developed for trajectory modeling and generation in TGM (namely spatial-temporal aggregation, adaptive budget allocation, and directed weighting) outperform alternate techniques derived from prior works in literature.

We first describe the datasets and utility measures in Section 5.6.1. The empirical end-to-end scalability and utility results are presented in Section 5.6.2 followed by the utility evaluation on TGM components in Section 5.6.3.

5.6.1 Experimental Setup

In the experiments that follow, each configuration is repeated 5 times, and the average result is reported. The budget ratio r in the generation process of TGM, i.e., Algorithm 9 in Section 5.4, is set as $r = 0.25$ and hence, $\varepsilon_1 = 0.25\varepsilon$ and $\varepsilon_2 = 0.75\varepsilon$. Unless otherwise specified, the window size for directed weighting is set to 10. We used a single machine with 32 CPUs of $2.30GHz$, $25MB$ cache and $64GM$ RAM. The default epsilon in all the experiments is set to 0.5.

Datasets. We consider two large datasets of trajectories in our experiments summarized in Table 5.1.

PORTO-TAXI. It describes a complete year (from 01/07/2013 to 30/06/2014) of the trajectories performed by 442 taxis running in the city of Porto, in Portugal. The dataset covers the region of Porto within a bounding box $(40.989727\,N, -8.793{,}681\,W)$ and $(41.465074\,N, -7.884{,}449\,W)$ — approximately 76 $km \times 52$ km. In this dataset, GPS points are sampled every 15 seconds and, unless otherwise specified, we set min-stay time $\lambda = 60$ seconds. This means that by starting from the first location in a trajectory and dividing the trajectory into groups of 4 points, the second, third and forth points in each group are eliminated due to our temporal aggregation defined in Section 5.3. Considering the average velocity as 7 m/s (40 km/h) for the moving objects, the

covering area is discretized into regions of size $(\omega \times \lambda)^2 = (7 \times 60)^2 m^2$. The average stay time (st_{avg}), average length (l_{avg}) and maximum length (l_{max}) of trajectories are reported in 5.1.

MELB-CAR. It is a synthetic dataset of trajectories generated by the Minnesota Web-based Traffic Generator [75] over the city of Melbourne, Australia. This dataset covers the region of Melbourne within a bounding box $(-38.467{,}210\ S, 144.572917\ E)$ and $(-37.516{,}525\ S,$ $145.449117\ E)$ — approximately 76 $km \times 105\ km$. The data is sampled every 20 seconds. Unless otherwise specified, we considered $\lambda = 120$ seconds for this dataset. Considering the average velocity as 11 m/s ($40km/h$) we discretized the covering area into regions of size $(\omega \times \lambda)^2 = (11 \times 120)^2 m^2$. Without loss of generality, we consider each trajectory representing a unique moving object in both datasets.

Utility measures. We analyze the utility of generated trajectories from three key dimensions: travelled distance, stay locations and frequent patterns.

Travelled distance (Q_d): For a trajectory $T = \{s_1, ..., s_n\}$, the travelled distance is defined as the maximum distance between any pair of locations in T, i.e., $max_{i,j} d(s_i, s_j) \forall 1 \leq i, j \leq n$. Given a trajectory dataset D, $Q_d(D)$ measures the distribution of travelled distance for all trajectories in D. The travelled distances are quantized into 20 bins of equal width each representing 1000 meters in POTO-TAXI, and 40 bins of 2500 meters width in MELB-CAR datasets. The utility of travelled distance in the synthetic dataset D_{syn} comparing to the raw dataset D_{raw} is measured by:

$$\mathscr{U}_{dist} = JSD(Q(D_{syn}), Q(D_{raw})),$$

where $JSD(\cdot)$ is the Jensen Shannon divergence with range $[0,1]$. If the distribution of travelled distance in D_{syn} is close to the distribution in D_{raw},i.e., $JSD(\cdot)$ value is close to zero, it shows a high utility of synthetic dataset in maintaining this property. The trajectories in D_{syn} might not exactly have the same behaviour of raw trajectories in D_{raw}. $JSD(\cdot)$ allows having instances that are in D_{raw} but not in D_{syn} and vice versa.

Frequent patterns (Q_p): We map a trajectory to a uniform grid according to our aggregation strategy in Section 5.3 and represent each location point by the centroid of the corresponding region. If multiple consecutive points located in one cell, we only consider one of them in the transformed trajectory. Let D be a dataset where a trajectory is represented by a sequence of

centroids. $Q_p(D)$ computes top k patterns in the dataset. We set k as 1000 in our experiments and measure the utility of D_{syn} against D_{raw} by :

$$\mathscr{U}_{patt} = F_1(Q_p(D_{syn}), Q_p(D_{raw})),$$

where $F_1(\cdot)$ is F_1 score with range $[0,1]$ and measures the similarity between item sets by a harmonic mean of precision and recall. The synthetic trajectories generated by TGM may pass through regions that are not visited by raw trajectories. We utilize two approximate match metrics to compute the distance of synthetic versus raw trajectories:

- Frechet distance that highlights the worst cases by considering the maximum Euclidean distance of points as the distance of two patterns.
- Dynamic Time Wrapping (DTW) distance that reports the overall quality of a pattern by computing the accumulated Euclidean distance of points in two patterns.

In computing the distances, two points are considered to be matched if they are both in the same region or located in adjacent regions.

Stay time (Q_s): We map a trajectory to a uniform grid according to our aggregation strategy in Section 5.3 and represent each location point by the centroid of the corresponding region. However, in contrast to the mapping used in the frequent pattern measure, we take into account all points in a trajectory, even those locating in one region. Let D be a dataset where a trajectory is represented as a sequence of adjacent centroids (not necessarily distinct). $Q_s(D)$ measures the distribution of total stay time for all trajectories in D. The stay times are quantized into 20 bins of equal width where the width is identified based on the $[min, max]$ stay time interval in D_{raw}. The utility of stay time in D_{syn} against D_{raw} is measured by:

$$\mathscr{U}_{dist} = JSD(Q(D_{syn}), Q(D_{raw})).$$

5.6.2 TGM Performance Evaluation

We compare the performance of TGM with the state-of-the-art algorithm, DPT [20]. DPT is a recent work proposed for publishing trajectories and improved the performance of prior algorithms [18] [19] [21]. This algorithm constructs a hierarchy of uniform grids using a given range

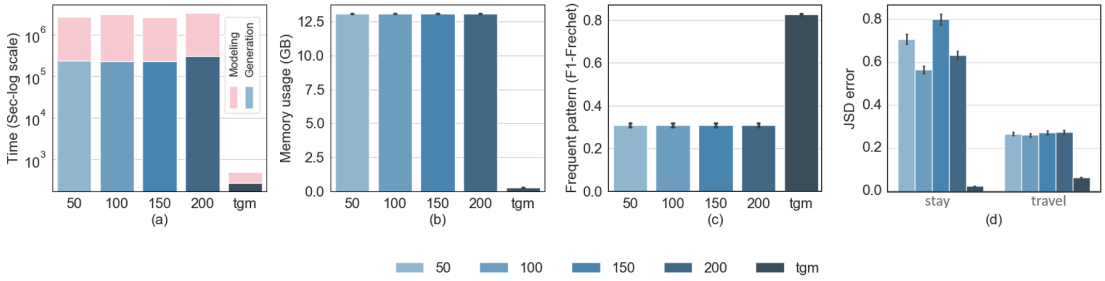


Figure 5.4: Performance of TGM versus DPT by varying the length (i.e., number of points) of synthetic trajectories in DPT. TGM does not need trajectory lengths and its performance does not depend on lengths.

of resolutions to capture changes in the movement speeds of objects. Each grid represents a discretization of the spatial space with a particular resolution derived from a specific speed. After mapping the trajectories into a grid, DPT builds a prefix tree to maintain the statistics of regions in that resolution. At each grid, if multiple consecutive points from a trajectory locate in a single region, DPT considers all of them as a single point represented by the centroid of that region. Given the constructed prefix trees, DPT adds noise to the values and prunes the noisy trees to ensure consistency. Next, in the generation step, DPT searches through all the prefix trees and uses Markov properties to generate synthetic trajectories of a specified length.

In our evaluations, unless otherwise specified, we set the height of prefix trees as 8, the speed range as $[100, 200, 400, 1600, 3200]$, the length of synthetic trajectories as 100 and the window size as 15 for DPT algorithm. Also, the budget ratio for selecting the best trees is set as 0.01 in the experiments. The reported results are the average of 5 repeated executions of the algorithm.

Scalability. Figure 5.4 shows the performance of TGM versus DPT on a sample of 100,000 trajectories from PORTO-TAXI dataset. DPT requires the length of synthetic trajectories to be specified by the user. Choosing a short length may cause missing the longer trajectories while a long length may lead to degrading the utility of generated trajectories. TGM generation process, in contrast, can dynamically identify the length of a synthetic trajectory. Figure 5.4 (a) and (b) evaluate the running time of DPT in seconds (log scale) and the memory usage in gigabytes in a selected length compared to TGM. The consumed memory by DPT is 13.074 *GB*, which is considerably larger than TGM that takes 0.27 *GB*. This memory usage is fixed for the selected lengths. The reason is that DPT *always* indexes the *entire* spatial space and constructs multiple tree structures which all should be maintained in the memory. Although the algorithm selects

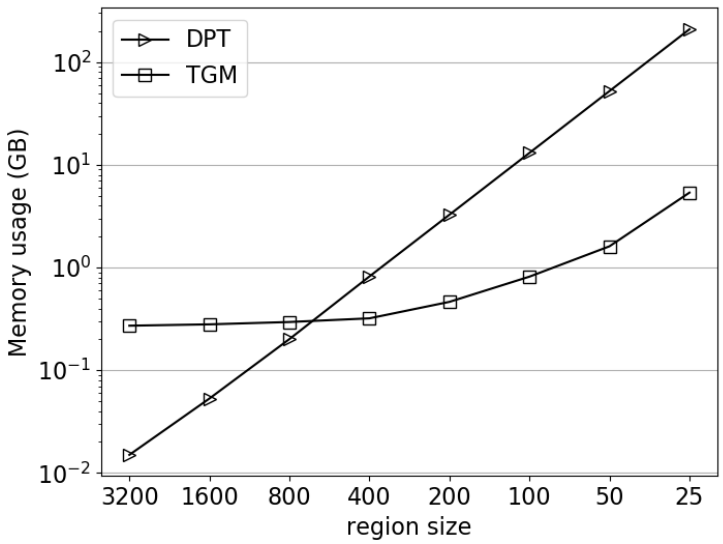


Figure 5.5: The growth of memory usage by increasing the resolution of grid.

some of the trees to reduce the memory usage and computation cost, we observe that the memory usage of DPT is still very high. TGM, instead of indexing the entire space, focuses on the regions that contain information. In particular, if no trajectory has passed through a region, the region would not be added to the TGM graphical model. This elegant feature leads to the significant improvement of memory usage in TGM. Note that the generation process in TGM still allows selecting such regions as they are part of the spatial space with some probability to be selected. The running times are shown in Figure 5.4 (a) depict the notable efficiency of TGM due to the indexing strategy. The modeling and then the generation step of TGM efficiently complete in 272 and 233 *seconds*, respectively. DPT in contrast, needs approximately 235,000 seconds ($\sim$ 65 hours) for modeling and 2,425,000 seconds ($\sim$ 673 hours) for only generating 100,000 synthetic trajectories which makes the algorithm hardly practical in machines with general configurations. The efficiency of TGM is because of two key feature of algorithms: (1) taking only the informative regions in the modeling step and (2) ability of the algorithm to generate synthetic trajectories in parallel. In DPT both steps of modeling and generation require searches through the space to compute required statistics. Additionally, selecting the next location of a synthetic trajectory needs searching n-grams through all the trees, which causes a considerable running time for the generation step compared to the model construction. We see that the running time of DPT is not

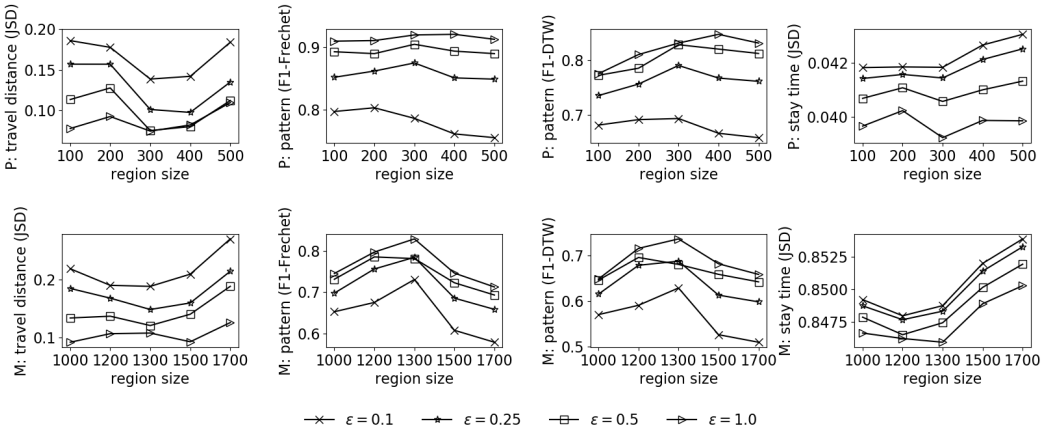


Figure 5.6: Spatial-temporal aggregation.

stable to the length of trajectories. This may happen due to the pruning step that decreases the search space of the algorithm.

Before comparing the utility of two algorithms, we discuss the effect of resolution on the efficiency of algorithms. The resolution of all uniform grids in DPT should be identified by the user. Increasing the resolution makes smaller regions but larger number of regions to be captured in the model. The intuition of taking a range of resolutions in DPT is to capture changes in the speed of moving objects. However, it is not clear how the user should decide about a "good" range of resolutions for a given dataset. In PORTO-TAXI dataset for example, one may consider the average speed of vehicles (any type) in Porto which a range $50-60$ km/h or a smaller range $(20-30)$ km/h to consider the effect of expected extra brakes for taxis. However using such fine resolutions, the memory usage of DPT increases exponentially, which cause the running time to increase beyond multiple days. In Figure 5.5 we contrast the growth of memory usage in a gigabyte (log scale) by TGM and DPT when the region size decreases — i.e., increasing resolution. The memory required by TGM nearly increases linearly to the resolution of the grid while this increase in DPT is exponential. As mentioned, this behaviour refers to the effective strategy of TGM in the modeling step. Since TGM maintains different statistics about the corresponding regions, we observe that in larger region sizes (lower resolutions) the memory usage of TGM slightly exceeds DPT.

Utility. Figure 5.4 (c) and (d) evaluate the utility of generated trajectories and the effect of selected length in DPT. We observe that TGM significantly outperforms DPT in different utility

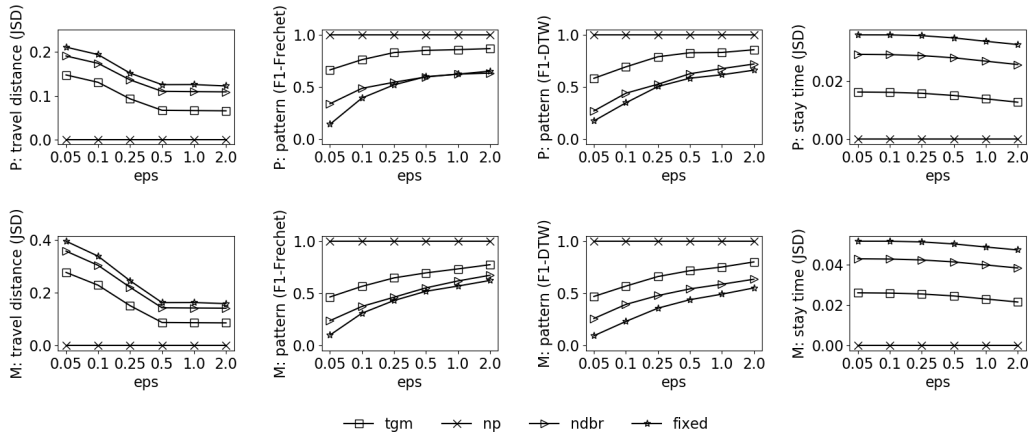


Figure 5.7: Directed weighting in generation process.

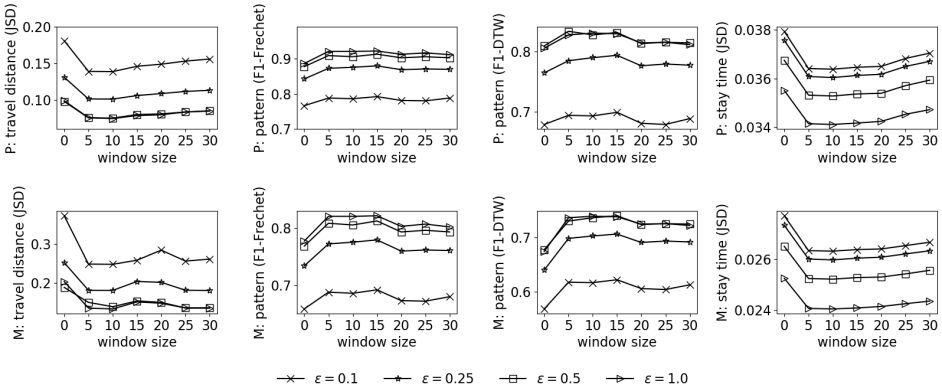


Figure 5.8: Directed weighting in generation process.

measures. For DPT, it is expected to observe changes in the utility by changing the length of trajectories. However, according to the results, the generated trajectories almost capture the same patterns in the data with nearly similar travelled distance. Considering this with the results of the stay time measure, we can realize the effect of length on the utility of DPT. As Figure 5.4 (d) shows, TGM significantly outperforms DPT in maintaining the stay times in the generated trajectories by achieving 0.023 JSD error. DPT fails in preserving the stay times its performance considerably changes by varying the length. Adding this to the stable utility in other measures, it implies that the generation process in DPT selects many points from a single cell. In particular, changing the length does not push the algorithm to investigate new regions or paths, but instead, it may make the algorithm to select more points from currently visited regions. The inconsistency

of this behaviour to the length can be because of the pruning step in DPT, which removes some nodes in the trees.

5.6.3 Component Utility Evaluation

In this section, we evaluate the effectiveness of our introduced techniques in each step of TGM: (1) spatial-temporal aggregation in modeling (2) dynamic budget allocation and restoring in the generation process, and (3) directed weighting strategy in the generation process. In the experiments, we measure the utility of generated trajectories using the measures discussed in Section 5.6.1. The results are shown on PORTO-TAXI and MELB-CAR datasets.

Component 1: Spatial-Temporal Aggregation. Figure 5.6 shows how the utility of TGM changes by varying the size of regions in the grid. Each graph reports the output given a specific privacy budget (ε). To generate the regions, we considered a fixed min-stay time (see Section 5.6.1) and changed the velocity is varied to make uniform grids of different resolutions for our spatial-temporal aggregations. The results confirm the effectiveness of our proposed strategy for aggregation as the utility of TGM is quite substantial on different measures. We observe a slight degradation in the utility of TGM on MELB-CAR compared to PORTO-TAXI, especially in higher resolutions (smaller region sizes). This result could be expected as this dataset covers a wider spatial area, and the simulated trajectories are much larger compared to PORTO-TAXI. TGM shows a stable utility across the region sizes and the effect of ε budget is negligible. The resulted graphs of frequent pattern similarity are nearly horizontal for Frechet distance showing that the maximum deviation in the generated trajectories is not highly sensitive to this grid resolution. For the accumulated deviation error (measure by DTW distance), we observe the utility increases for larger regions, especially in larger ε values. Additionally, the utility in preserving the stay locations increases in larger regions. This behaviour highlights the importance of considering data content in choosing the min-stay time and velocity for spatial-temporal aggregation.

Component 2: Budget Allocation. Figure 5.7 analyses the effectiveness of using adaptive budget allocation at each step of generation as well as saving ε usage based on our direction budget restoring strategy. In the figure, *np* represents the optimal result with no privacy noise as a baseline. *fixed* refer to using a fixed portion of ε for each step of generating a synthetic trajectory. *ndbr* uses the TGM settings except the directed budget restoring for saving budget in some step

of generation. The higher slope of utility improvement in the corresponding graph of *fixed* and *ndbr* approaches to reveal the sensitivity of these approaches to the added noise and hence, lack of robustness. The results show that TGM is a promising approach on both datasets, even in stingy privacy budgets. We observe that using adaptive budget allocation improve the utility of generated trajectories comparing to fixed budget allocation. However, the significant improvement happens when the budget restoring strategy is applied. We mainly observe the usefulness of this technique on MELB-CAR dataset where longer trajectories should be constructed.

Component 3: Directed Weighting. The effect of window size on the utility of generated trajectories by TGM is shown in Figure 5.8. A small window cannot capture the required information to improve the generation process. On the other hand, large windows size takes into account the number of recent movements that again may not provide accurate information about the movement direction of objects. A window size of zero means not using this technique in the generation process. As the results show, a window size of 5 to 10, recent movements would improve the performance of TGM. The usefulness of using this technique is evident in capturing the stay time when generating synthetic trajectories. We also observe the effectiveness of windowing in maintaining the frequent patterns as it significantly reduces the total deviation of synthetic trajectories measured by DTW distance.

5.7 Conclusion

Trajectory Generative Mechanism (TGM) is a two-stage algorithm for publishing trajectory datasets with differential privacy guarantee. TGM first transforms the data into a data-adaptive aggregated space and constructs a graphical generative model to capture the statistics of trajectories. Our experiments show that TGM is highly memory and time efficient, especially when the data covers a large metropolitan area. The second step privately generates a synthetic dataset of trajectories that maintain the key properties of the original dataset. We introduce a set of techniques in this step that makes the process data-adaptive and hence, it produces trajectories without needing any assumptions about trajectory properties such as length, velocity or directions of movement.

Our results show that applying the data properties in the modeling and the generation process can significantly improve the efficiency of the algorithm and the utility of generated dataset. In our

future work, we incorporate the information of the underlying metropolitan area and road networks and pruning a large number of regions of low trajectory density which can result in significant improvement in efficiency in the encoding step as well as accuracy in the generation step of the algorithm. We view this as a decisive direction. We would also like to consider extensions of our aggregation strategy that would incorporate the exact timestamp of location points along with their relative time and extend our algorithm to publish trajectories with exact temporal features.

Chapter 6
Conclusion

6.1 Summary of Contributions

IN this thesis, we developed mechanisms for publishing trajectory datasets under differential privacy. We specifically focused on the utility of published data as the key to the success of mechanisms. By ensuring differential privacy and utility, the mechanisms guarantee preserving individual privacy as well as high utility for spatial analytic on the published data to third parties. Our mechanisms can be applied in various applications, including urban planning, location-based marketing, targeted advertising, behavioural profiling, transportation analysis, etc. The core question of this thesis is how a privacy-preserving solution can ensure high utility while providing differential privacy as a proven guarantee on individual privacy. We could show in this thesis that the added noise to satisfy differential privacy can be customized using domain-specific knowledge (integrity constraints and query properties) which leads to achieving high utility by the mechanism. In this chapter, we summarize the findings made throughout the thesis and discuss future research directions.

6.2 Summary of the Research Contributions

In this section, we summarize the main goal, challenges and contributions of each core chapter.

6.2.1 Privacy-Preserving Processing of Trajectories Using Aggregated Data

In Chapter 3, we propose PriSH, a mechanism for publishing spatial histograms under differential privacy guarantees. Given a trusted data collector, a spatial histogram can be built in two settings:

i) users send their trajectory to the data collector who publishes the aggregated counts; ii) users only transmit and update the count of their current location. The published histogram can then be used in computing range queries wherein the number of distinct trajectories in a specified range is computed. The key insight is to exploit the given set of range queries in estimating the distribution of trajectory counts throughout the spatial space and then, using the estimation to generate a synthetic spatial histogram. Perturbing the synthesizing process using the Laplace mechanism and Exponential mechanism ensures preserving individual privacy.

The main challenge, however, is maintaining the utility of published histogram. The fact that a trajectory may overlap multiple cells in the histogram requires an advanced approach that carefully calibrates the noise to the histogram properties.

In our work, we formulate the sequential dependency of cells and the consistency constraints in each cell to be preserved. Our mechanism utilizes the data dependency in calibrating the scale of added noise. PriSH also ensures the consistency of cell values in the published histogram. We provide a theoretical bound on the utility of published histogram by PrisH that evaluates the worst-case scenario in answering range queries. Our extensive experimental analysis demonstrates the significant utility of published histogram by PriSH in answering range queries compared with the state-of-the-art approaches.

6.2.2 Optimal Spatial Histograms Under Differential Privacy

Whilst PriSH achieves high utility in publishing spatial histograms, the performance of a mechanism is sensitive to the underlying data distribution. In particular, the synthesizing process in PriSH is not data adaptive to capture congested versus sparse areas in estimating the trajectory counts. In Chapter 4, we propose DQAM, which is a data- and query-aware mechanism for privately publishing spatial histograms. The synthesizing process in DQAM employs the given query set as well as the information computed about the data distribution. The main insight is to estimate the trajectory counts of each area with respect to its congestion and hence, enable the mechanism to capture an accurate estimation of data distribution even when it is a skewed distribution. Our mechanism also computes the optimal, consistent estimation that ensures all the consistency constraints in a spatial histogram are satisfied.

The main challenge is efficiently computing the information about the trajectories' congestion

throughout the spatial space. There is no closed-form for computing congestion-based partitions, and the computational complexity grows with an exponential order.

In our work, we estimate the congestion-based partitioning by finding uniform partitions in the data. Our mechanism employs an efficient search structure to compute the information with $nlog(n)$ complexity in the number of cells in the spatial histogram.

We provide the upper bound of utility for the synthesized histogram by DQAM. The worst-case scenario is evaluated for congestion-based partitioning and computing the optimal, consistent estimation.

Our extensive experimental evaluations demonstrate substantial improvement in the utility of published histogram by considering both data and query information. Our mechanism can accurately capture the trajectory counts in various distributions from uniform to highly skewed.

6.2.3 A Differentially Private Mechanism for Publishing Trajectories

Aggregated statistics are usually developed for a particular query type on trajectory datasets. Spatial histogram, as an instance of aggregated statistics on trajectory datasets, is restricted to range queries only while many applications can be based on other query types. Finding frequent patterns in trajectory movements, computing total travel distance or detecting locations of interest in a trajectory are instances that cannot be addressed given some aggregated statistics. In chapter 5, we introduce TGM for publishing trajectory datasets under differential privacy, which can be used for any analytical task. Our mechanism employs a graph-based structure to capture the statistics about the movement behaviour of trajectories accurately and uses the statistics to generate synthetic trajectories under differential privacy. The key insight is that i) a graph structure allows capturing details of trajectories movement with no restriction on their movement patterns or traveled distance, and ii) randomizing the generation process enables the mechanism to generate trajectories of various patterns and lengths. The main challenge in TGM is ensuring the utility of generated trajectories considering the limited privacy budget for each trajectory.

In our mechanism, we propose an adaptive budget allocation technique for generating synthetic trajectories wherein the budget is adjusted to the information gained about the target trajectory throughout the generation process. We also introduce a budget restoring technique that enables our mechanism to save privacy budget in some stages of the generation process and hence, achieve

synthetic trajectories of higher utility.

In this chapter, we have shown that it is indeed possible to achieve a high accuracy for publishing trajectories under differential privacy. Our theoretical and extensive empirical analysis demonstrate the significant performance of our mechanism in contrast to the state-of-the-art mechanisms including efficiency in running time and memory usage as well as accuracy in preserving frequent movement patterns, total travel distance and locations of interest in trajectories.

6.3 Future Research Directions

The research conducted in this thesis suggests a range of new directions that could advance the field of privacy-preserving trajectory publishing.

6.3.1 Integrating Exact Timestamps of Trajectories

In analyzing trajectories, time is often seen as less importance than the footprint of samples. However, in most real-time applications such as route planning, flow control and navigation, and prediction of traffic jam, explicit incorporation of time might be necessary. The application might need to analyze movement behaviours in various time frames, including per hour, per day, per month, etc.

Integrating exact timestamps for publishing trajectories is an open research problem. This entails adding the time parameter to the process of modeling and then generating synthetic trajectories which in turn adds new complexities to the mechanism. Similar to the location, time is taken from a continuous domain. Considering a graphical model such as the one we proposed in this thesis, incorporating raw timestamps in each graph node entails a notable increase in the mechanisms' memory usage. Moreover, considering raw time instances may affect the trajectory generation as hardly two trajectories make a move in exactly the same timestamp. Thus, a temporal aggregation strategy should be developed to capture the movements in trajectories accurately. The aggregated timestamps need to be incorporated in the graphical model. One strategy is to consider the temporal component as a new feature at each node of the graphical model. Another strategy is to consider two node types for the graphical model: spatial node and temporal node. The advantage of this strategy is that it allows accurately modeling the temporal component of trajectories

that might be in different spatial regions but have happened in the same time period. Also, this strategy saves the memory usage by avoiding redundant times recorded in graph nodes. Constructing synthetic trajectories, however, is more challenging with the two node types as the exponential mechanism has more options for randomization. The utility function in the exponential mechanism should be designed carefully to maintain the utility of generated trajectories.

6.3.2 Lower Utility Bounds

An open research problem in publishing trajectory datasets is computing the lower bound of utility for synthetic trajectories. In this thesis, we demonstrated the utility of trajectories generated by our mechanism through extensive empirical analysis. However, a theoretical lower bound on the utility would provide a measure in evaluating the efficacy of trajectory publishing mechanisms. In real applications, some datasets might not be amenable to differential privacy as the introduced noise might lead to no useful utility. In a sparse trajectory dataset, for example, where each trajectory is unique in a particular area, ensuring differential privacy requires adding a considerable amount of noise to each trajectory to hide it among the other ones in the dataset. In particular, as the degree of uniqueness of individual records is higher in a dataset (heterogeneous), it is more sensitive to noise and vice versa. It is imperative to identify which datasets are amenable to privatization, i.e., tolerating a level of noise.

6.3.3 Multi-Granularity in Data Publishing

In a trajectory dataset, the moving objects generating trajectory data do not necessarily move with a constant speed. In a taxi trajectory dataset, for example, a taxi may drive with higher speeds on highways but has to drive slower in the city center or crowded areas. Developing a private mechanism that captures such changes in the movement speed while preserving the utility of trajectories is important. The speed parameter can be considered as a factor in aggregating the spatial and temporal domains. Let the sampling rate of data be fixed. When the moving object has a slow speed, the aggregated area should be small to capture all the sampled locations. For a fast-moving object, however, large aggregated areas are better choices as they accurately capture the distance between two consecutive sampled points. Thus, developing an aggregation strategy that

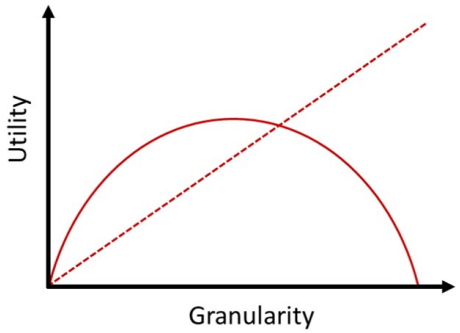


Figure 6.1: The effect of granularity on trajectories utility. Dashed line depicts the effect without noise in data. The curve line show the effect when noise is added due to privacy.

accurately captures changes in speed as well as stay locations in movement patterns of trajectories is of great importance as it aligns more with the nature of real datasets.

Identifying the optimal granularity or hierarchy of granularities for aggregation while preserving the data utility is non-trivial, especially when the aggregated data is perturbed by noise due to differential privacy. To describe the challenges, assume the speed of moving objects is fixed in the dataset. In this case, increasing the granularity of data leads to increasing the utility of trajectories as more samples or more fine-grained trajectory data is provided. However, when noise is added to data, this relation will no longer be linear. Increasing the granularity leads to fewer trajectory counts per grid cell and hence, more sensitivity to noise. Figure 6.1 shows an example relation of granularity versus utility in these two scenarios. To compute the optimal granularity, both the data properties and the effect of noise by changing the granularity should be considered. Taking into account the changes in speed will add another parameter to this problem that makes it challenging.

6.3.4 Noise Distributions Compatible with Spatio-Temporal Data

Basic mechanisms to achieve differential privacy, i.e., the Laplace mechanism and the exponential mechanism, were initially designed for relational datasets. The Laplace mechanism specifically was developed for continuous numerical data while the exponential mechanism was introduced for tackling with categorical data. The underlying assumption in both mechanisms is the independency of features in data as they add a random noise driven from the specified distribution to each feature independently. In most of real applications, however, there is a dependency between features. Researchers later employed these basic mechanisms in developing advanced algorithms

to consider dependency of features while scaling the added noise. Depending on the data type, the algorithms apply aggregation or transformation on data to reduce the dependency and hence, the noise level. Whilst the algorithms successfully achieve a reasonable utility level in many applications, they make some assumptions about the data that degrades the applicability of generated output. For example, locations in a trajectory are sequentially dependent, and a trajectory may be arbitrarily long with no particular movement patterns. Due to the dependency of locations, a naive mechanism should scale the noise to the number of locations in a trajectory which significantly affect the utility of output trajectories. The proposed algorithms make assumptions about the length or movement pattern of trajectories. Such assumptions enable the algorithms to map trajectories into a structure with lower sensitivity to noise based on the basic mechanism, i.e., the Laplace and the exponential mechanism. Due to such challenges, many studies on publishing trajectories do not consider the temporal component in modeling the data. In addition, both temporal and spatial information can be highly correlated. For example, going to school at 8:00 am is highly likely and does not require perturbing either spatial or temporal information. Whereas going to a cafe might need perturbation of both location and time. We need differential privacy mechanisms that take into consideration such domain knowledge.

Bibliography

[1] A. Hern. (2018) Fitness tracking app strava gives away location of secret us army bases. [Online]. Available: TheGuardian

[2] P. Vines, F. Roesner, and T. Kohno, "Exploring ADINT: using ad targeting for surveillance on a budget - or - how alice can buy ads to track bob," in *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, Dallas, TX, USA, October 30 - November 3, 2017*. ACM, 2017, pp. 153–164.

[3] B. X. Chen. (2011) Why and how apple is collecting your iphone location data. [Online]. Available: WIRED

[4] J. Valentino-DeVries, N. Singer, M. H. Keller, and A. Krolik. (2018) Your apps know where you were last night, and theyre not keeping it secret. [Online]. Available: TheNewYorkTimes

[5] F. Xu, Z. Tu, Y. Li, P. Zhang, X. Fu, and D. Jin, "Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1241–1250.

[6] Z. Tu, F. Xu, Y. Li, P. Zhang, and D. Jin, "A new privacy breach: User trajectory recovery from aggregated mobility data," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1446–1459, 2018.

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith, *Calibrating Noise to Sensitivity in Private Data Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284.

[8] L. Hay Newman. (2019) Google wants to help tech companies know less about you. [Online]. Available: WIRED

[9] A. Greenberg. (2016) Apples differential privacy is about collecting your databut not your data. [Online]. Available: WIRED

[10] K. Tezapsidis. (2017) Uber releases open source project for differential privacy. [Online]. Available: Medium

[11] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias, "Spatio-temporal aggregation using sketches," in *Data Engineering Proceedings. 20th International Conference on*. IEEE, 2004, pp. 214–225.

[12] I. V. Lopez, R. T. Snodgrass, and B. Moon, "Spatiotemporal aggregate computation: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 271–286, 2005.

[13] L. Leonardi, S. Orlando, A. Raffaetà, A. Roncato, C. Silvestri, G. Andrienko, and N. Andrienko, "A general framework for trajectory data warehousing and visual olap," *GeoInformatica*, vol. 18, no. 2, pp. 273–312, 2014.

[14] H. Xie, E. Tanin, and L. Kulik, "Distributed histograms for processing aggregate data from moving objects," in *Mobile Data Management, 2007 International Conference on*. IEEE, 2007, pp. 152–157.

[15] H. Xie, E. Tanin, L. Kulik, P. Scheuermann, G. Trajcevski, and M. Fanaeepour, "Euler histogram tree: A spatial data structure for aggregate range queries on vehicle trajectories," in *Proceedings of the 7th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. ACM, 2014, pp. 18–24.

[16] M. Fanaeepour, L. Kulik, E. Tanin, and B. I. Rubinstein, "The case histogram: privacy-aware processing of trajectory data using aggregates," *GeoInformatica*, vol. 19, no. 4, pp. 747–798, 2015.

[17] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*. IEEE, 2007, pp. 94–103.

[18] R. Chen, G. Acs, and C. Castelluccia, "Differentially private sequential data publication via variable-length n-grams," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 638–649.

[19] L. Bonomi and L. Xiong, "A two-phase algorithm for mining sequential patterns with differential privacy," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013, pp. 269–278.

[20] X. He, G. Cormode, A. Machanavajjhala, C. M. Procopiuc, and D. Srivastava, "Dpt: differentially private trajectory synthesis using hierarchical reference systems," *Proceedings of the VLDB Endowment*, vol. 8, no. 11, pp. 1154–1165, 2015.

[21] R. Chen, B. Fung, B. C. Desai, and N. M. Sossou, "Differentially private transit data publication: a case study on the montreal transportation system," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 213–221.

[22] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[23] P. Lou, G. Zhao, X. Qian, H. Wang, and X. Hou, "Schedule a rich sentimental travel via sentimental poi mining and recommendation," in *2016 IEEE second international conference on multimedia big data (BigMM)*. IEEE, 2016, pp. 33–40.

[24] X. Liu, Y. Liu, K. Aberer, and C. Miao, "Personalized point-of-interest recommendation by mining users' preference transition," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. ACM, 2013, pp. 733–738.

[25] B. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Computing Surveys (CSUR)*, vol. 42, no. 4, p. 14, 2010.

[26] L. Burnett, K. Barlow-Stewart, A. Proos, and H. Aizenberg, "The genetrustee: a universal identification system that ensures privacy and confidentiality for human genetic databases," *Journal of Law and Medicine*, vol. 10, no. 4, pp. 506–513, 2003.

[27] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 571–588, 2002.

[28] P. Samarati and L. Sweeney, "Generalizing data to provide anonymity when disclosing information (abstract)," in *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, 1998, p. 188.

[29] M. E. Nergiz, C. Clifton, and A. E. Nergiz, "Multirelational k-anonymity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 8, pp. 1104–1117, 2009.

[30] K. Wang and B. Fung, "Anonymizing sequential releases," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 414–423.

[31] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 3, 2007.

[32] T. M. Truta and B. Vinay, "Privacy protection: p-sensitive k-anonymity property," in *Proceedings of the 22nd International Conference on Data Engineering Workshops, ICDE 2006, 3-7 April 2006, Atlanta, GA, USA*, 2006, p. 94.

[33] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *2007 IEEE 23rd International Conference on Data Engineering*. IEEE, 2007, pp. 106–115.

[34] J. Li, Y. Tao, and X. Xiao, "Preservation of proximity privacy in publishing numerical sensitive data," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 2008, pp. 473–486.

[35] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang, "(α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 754–759.

[36] M. E. Nergiz, M. Atzori, and C. Clifton, "Hiding the presence of individuals from shared databases," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 665–676.

[37] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar, "Random-data perturbation techniques and privacy-preserving data mining," *Knowledge and Information Systems*, vol. 7, no. 4, pp. 387–414, 2005.

[38] V. Rastogi, D. Suciu, and S. Hong, "The boundary between privacy and utility in data publishing," in *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007, pp. 531–542.

[39] C. Dwork, "A firm foundation for private data analysis," *Communications of the ACM*, vol. 54, no. 1, pp. 86–95, 2011.

[40] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference*. Springer, 2006, pp. 265–284.

[41] F. D. McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 19–30.

[42] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*. IEEE, 2007, pp. 94–103.

[43] V. Rastogi and S. Nath, "Differentially private aggregation of distributed time-series with transformation and encryption," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM, 2010, pp. 735–746.

[44] G. Acs, C. Castelluccia, and R. Chen, "Differentially private histogram publishing through lossy compression," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 1–10.

[45] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 8, pp. 1200–1214, 2011.

[46] M. Hardt, K. Ligett, and F. McSherry, "A simple and practical algorithm for differentially private data release," in *Advances in Neural Information Processing Systems*, 2012, pp. 2339–2347.

[47] J. Xu, Z. Zhang, X. Xiao, Y. Yang, G. Yu, and M. Winslett, "Differentially private histogram publication," *The VLDB Journal*, vol. 22, no. 6, pp. 797–822, 2013.

[48] M. Hay, V. Rastogi, G. Miklau, and D. Suciu, "Boosting the accuracy of differentially private histograms through consistency," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1021–1032, 2010.

[49] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar, "Privacy, accuracy, and consistency too: a holistic solution to contingency table release," in *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2007, pp. 273–282.

[50] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor, "Optimizing linear counting queries under differential privacy," in *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2010, pp. 123–134.

[51] C. Li, M. Hay, G. Miklau, and Y. Wang, "A data-and workload-aware algorithm for range queries under differential privacy," *Proceedings of the VLDB Endowment*, vol. 7, no. 5, pp. 341–352, 2014.

[52] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan, "On the complexity of differentially private data release: efficient algorithms and hardness results," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*. ACM, 2009, pp. 381–390.

[53] C. Dwork, G. N. Rothblum, and S. Vadhan, "Boosting and differential privacy," in *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*. IEEE, 2010, pp. 51–60.

[54] L. Fan and L. Xiong, "An adaptive approach to real-time aggregate monitoring with differential privacy," *IEEE Transactions on knowledge and data engineering*, vol. 26, no. 9, pp. 2094–2106, 2013.

[55] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong, "Quantifying differential privacy under temporal correlations," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 821–832.

[56] P. Flajolet and G. N. Martin, "Probabilistic counting algorithms for data base applications," *Journal of computer and system sciences*, vol. 31, no. 2, pp. 182–209, 1985.

[57] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific reports*, vol. 3, p. 1376, 2013.

[58] M. Terrovitis, G. Poulis, N. Mamoulis, and S. Skiadopoulos, "Local suppression and splitting techniques for privacy preserving publication of trajectories," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 7, pp. 1466–1479, 2017.

[59] R. Beigel and E. Tanin, "The geometry of browsing," in *Latin American Symposium on Theoretical Informatics*. Springer, 1998, pp. 331–340.

[60] H. Xie, L. Kulik, and E. Tanin, "Privacy-aware traffic monitoring," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 61–70, 2010.

[61] R. Chen, B. C. Fung, N. Mohammed, B. C. Desai, and K. Wang, "Privacy-preserving trajectory data publishing by local suppression," *Information Sciences*, vol. 231, pp. 83–97, 2013.

[62] G. Poulis, S. Skiadopoulos, G. Loukides, and A. Gkoulalas-Divanis, "Apriori-based algorithms for kˆ m-anonymizing trajectory data," *Transactions on Data Privacy*, vol. 7, no. 2, pp. 165–194, 2014.

[63] E. Naghi Zadeh Kakhki, "Utility-aware protection of trajectory privacy," Ph.D. dissertation, The University of Melbourne, 2016.

[64] A. Monreale, W. H. Wang, F. Pratesi, S. Rinzivillo, D. Pedreschi, G. Andrienko, and N. Andrienko, "Privacy-preserving distributed movement data aggregation," in *Geographic Information Science at the Heart of Europe*. Springer, 2013, pp. 225–245.

[65] G. Cormode, C. Procopiuc, D. Srivastava, E. Shen, and T. Yu, "Differentially private spatial decompositions," in *Data engineering (ICDE), 2012 IEEE 28th international conference on*. IEEE, 2012, pp. 20–31.

[66] W. Qardaji, W. Yang, and N. Li, "Differentially private grids for geospatial data," in *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 2013, pp. 757–768.

[67] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang, "Principled evaluation of differentially private algorithms using dpbench," in *Proceedings of the 2016 International Conference on Management of Data*. ACM, 2016, pp. 139–154.

[68] K. Jiang, D. Shao, S. Bressan, T. Kister, and K.-L. Tan, "Publishing trajectories with differential privacy guarantees," in *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*. ACM, 2013, p. 12.

[69] S. Shang, L. Chen, C. S. Jensen, J.-R. Wen, and P. Kalnis, "Searching trajectories by regions of interest," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 7, pp. 1549–1562, 2017.

[70] Z. Huo, X. Meng, H. Hu, and Y. Huang, "You can walk alone: trajectory privacy-preserving through significant stays protection," in *International conference on database systems for advanced applications*. Springer, 2012, pp. 351–366.

[71] S.-S. Ho and S. Ruan, "Differential privacy for location pattern mining," in *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*. ACM, 2011, pp. 17–24.

[72] WalkScore. (2018) Melbourne is somewhat walkable. [Online]. Available: https://www.walkscore.com/AU-VIC/Melbourne

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

[74] M. Hardt and G. N. Rothblum, "A multiplicative weights mechanism for privacy-preserving data analysis," in *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*. IEEE, 2010, pp. 61–70.

[75] M. F. Mokbel, L. Alarabi, J. Bao, A. Eldawy, A. Magdy, M. Sarwat, E. Waytas, and S. Yackel, "Mntg: an extensible web-based traffic generator," in *International Symposium on Spatial and Temporal Databases*. Springer, 2013, pp. 38–55.

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

[77] IEEE. (2015) P2413 iot standard. [Online]. Available: https://standards.ieee.org/develop/project/2413.html

[78] J. S. Kumar and D. R. Patel, "A survey on internet of things: Security and privacy issues," *International Journal of Computer Applications*, vol. 90, no. 11, 2014.

[79] J. Joy and M. Gerla, "Internet of vehicles and autonomous connected car-privacy and security issues," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp. 1–9.

[80] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 1991.

[81] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[82] G. Atluri, A. Karpatne, and V. Kumar, "Spatio-temporal data mining: A survey of problems and methods," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, p. 83, 2018.

[83] M. Das and S. K. Ghosh, "Deep-step: A deep learning approach for spatiotemporal prediction of remote sensing data," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 12, pp. 1984–1988, 2016.

[84] J. R. Norris, *Markov chains*. Cambridge university press, 1998, no. 2.

[85] M. Lyu, D. Su, and N. Li, "Understanding the sparse vector technique for differential privacy," *Proceedings of the VLDB Endowment*, vol. 10, no. 6, pp. 637–648, 2017.