Answering Spatial Density Queries Under Local Differential Privacy

by

Ekin Tire

A Thesis Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

September 7, 2022

Answering Spatial Density Queries Under Local Differential Privacy

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Ekin Tire

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assist. Prof. Dr. Mehmet Emre Gürsoy (Advisor)

_____

Assoc. Prof. Dr. Alptekin Küpçü

_____

Prof. Dr. Yücel Saygın

Date: _____

to Mom and Dad

ABSTRACT

Answering Spatial Density Queries Under Local Differential Privacy
Ekin Tire
Master of Science in Computer Science and Engineering
September 7, 2022

Spatial density queries are fundamental in many geospatial data analysis tasks and have numerous applications in the real world, such as determining crowded areas, estimating traffic density, navigation, passenger demand analysis, and so forth. However, answering spatial density queries based on users' data may violate users' privacy by exposing their true locations to an untrusted third party (e.g., a server acting as the service provider or data collector).

In this thesis, we propose a solution for answering spatial density queries while preserving Local Differential Privacy (LDP), a state-of-the-art privacy protection standard. Our solution consists of four main steps: partitioning, finding sensitivity, user-side noisy response computation, and server-side estimation. For the first step, we initially propose three basic partitioning strategies: Singleton Partitioning, Holistic Partitioning and Random Partitioning. Based on our qualitative and empirical analysis of the three basic strategies, we design and implement an improved strategy called *Advanced Partitioning*. For the second step, we adapt graph-based modeling of query sets from the centralized DP literature. Advanced Partitioning also leverages and extends this technique by formulating the partitioning problem as a vertex coloring problem on the graph representation of a query set.

For the third and fourth steps, in addition to adapting two popular LDP protocols (GRR and RAPPOR) to our solution, we propose an extension for the Optimized Unary Encoding (OUE) protocol so that it can be employed in our solution. We call the extended protocol Optimized Bitvector Encoding (OBE). OBE is applicable to not only the problem of answering spatial density queries, but also in arbitrary LDP problems with bitvector encodings. We formally prove that the user-side perturbation step of our OBE protocol satisfies LDP and its server-side estimation step produces unbiased estimates.

Combining the different partitioning strategies and LDP protocols, we obtain a

total of 8 different approaches for answering spatial density queries under LDP, all of which can be parsed as instances of our four-step solution with different choices in the individual steps. We perform an extensive experimental evaluation of these approaches using 4 real-world datasets, varying number of queries, varying query sizes, varying degrees of privacy, and multiple error metrics. Results show that Advanced Partitioning and OBE protocol typically yield the lowest error, demonstrating the superiority of our proposed methods.

ÖZETÇE

Konumsal Yoğunluk Sorgularının Lokal Diferansiyel Mahremiyet Korumalı Cevaplanması

Ekin Tire

Bilgisayar Mühendisliği, Yüksek Lisans

7 Eylül 2022

Konumsal yoğunluk sorguları, birçok coğrafi veri analizi işinde temel bir yapıtaşı olmakla birlikte, kalabalık alanların belirlenmesi, trafik yoğunluğunun tahmini, navigasyon, yolcu talep analizi gibi gerçek hayatta çok sayıda uygulama alanına sahiptir. Öte yandan, konumsal yoğunluk sorgularının kullanıcı verileri kullanılarak cevaplanması, kullanıcıların gerçek konumlarını güvenilir olmayan üçüncü kişilere (örneğin, servis sağlayıcı veya veri toplayıcı görevi gören bir sunucuya) ifşa ederek kullanıcı mahremiyetini ihlal edebilir.

Bu tezde, konumsal yoğunluk sorgularının modern bir mahremiyet koruma standardı olan Lokal Diferansiyel Mahremiyet (LDP) korumalı cevaplanması için bir çözüm öneriyoruz. Çözümümüz dört ana adımdan oluşmaktadır: bölümleme, hassasiyet bulma, kullanıcı tarafı gürültülü cevap hesaplaması ve sunucu tarafı tahminleme. İlk adım için, öncelikle üç temel bölümleme stratejisi öneriyoruz: Tekil Bölümleme, Bütünsel Bölümleme ve Rastgele Bölümleme. Üç temel bölümleme stratejisi üzerinde yaptığımız nitel ve deneysel analizlere dayanarak, *Gelişmiş Bölümleme* adlı iyileştirilmiş bir strateji öneriyor ve geliştiriyoruz. İkinci adım için, merkezi DP literatüründen sorgu kümelerinin çizge-tabanlı modellemesi tekniğini kullanıyoruz. Gelişmiş Bölümleme yöntemi de çizge-tabanlı modelleme tekniğini kullanmakta ve bölümleme problemini sorgu kümesinin çizge modeli üzerinde düğüm renklendirme problemi olarak çözümleyerek geliştirmektedir.

Üçüncü ve dördüncü adımlar için, çözümümüze iki popüler LDP protokolünü (GRR ve RAPPOR) uyarlamaya ek olarak, çözümümüze uygulanabilir hale gelmesi için Optimized Unary Encoding (OUE) protokolüne bir genişletme öneriyoruz. Genişletilmiş protokolümüzün ismi Optimized Bitvector Encoding (OBE)'dir. OBE sadece konumsal sorguların cevaplanması problemiyle sınırlı olmayıp, bitvektör kodlaması içeren herhangi bir LDP probleminde de kullanıma uygundur. OBE'nin kullanıcı

tarafı gürültülü cevap hesaplamasının LDP'yi sağladığını ve sunucu tarafı tahminlemesinin tarafsız olduğunu matematiksel olarak kanıtlıyoruz.

Farklı bölümleme stratejileri ve LDP protokollerinin kombinasyonları kullanılarak, konumsal sorguların LDP ile cevaplanması için 8 farklı yaklaşım elde edilmiştir; bu yaklaşımların hepsi dört-adımlı çözümümüzün adımlarında farklı seçimler yapılarak elde edilen örnekleri olarak incelenebilir. Elde edilen yaklaşımların kapsamlı deneysel değerlendirmesini, gerçek dünyadan alınmış 4 veri seti, değişen sayıda sorgu, değişen sayıda sorgu boyutu, değişen mahremiyet dereceleri ve birden fazla hata metriği kullanarak yapıyoruz. Sonuçlar, Gelişmiş Bölümleme stratejisi ve OBE protokolünün genelde en düşük hatayı verdiğini göstermekte, bu durum da önerdiğimiz yöntemlerin üstünlüğünü gözler önüne sermektedir.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincerest gratitude to my advisor and mentor *Assist. Prof. Dr. Mehmet Emre Gürsoy* for his guidance and support in every step of my Masters journey. It would not be possible for me to grow as a researcher and complete this work without him.

Special thanks to *Assoc. Prof. Dr. Alptekin Küpçü* from Koç University and *Prof. Dr. Yücel Saygın* from Sabancı University for being part of my thesis jury committee, and their invaluable comments and suggestions to improve this work.

I am also grateful to *Tolga Okur, Zafer Serbest* and *Murat Serhat Adalı* from Codevo. I could not have completed my Masters while working full-time without their continuous support and understanding.

I wish to thank my loving partner *Gülce* for her patience and support throughout my stressful times during Masters.

Most importantly, I would like to thank my mom and dad for being amazing parents and for everything they have done for me in my life.

Lastly, I thank the members of the SPADE Lab at Koç University and gratefully acknowledge the support by TUBITAK under project number 121E303.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

LDP	Local Differential Privacy
DP	Differential Privacy
RMSE	Root Mean Squared Error
GRR	Generalized Randomized Response
RAPPOR	Randomized Aggregatable Privacy-Preserving Ordinal Response
OUE	Optimized Unary Encoding
OBE	Optimized Bitvector Encoding

Chapter 1

INTRODUCTION

Local Differential Privacy (LDP) has recently emerged as a popular standard for privacy protection in distributed multi-user environments with an untrusted data collector [Cormode et al., 2018a, Thakurta et al., 2017, Ding et al., 2017, Erlingsson et al., 2014, Wang et al., 2017]. In LDP, each user perturbs their data on their local device before sending the perturbed output to the data collector. The data collector cannot infer the true value of a user from their perturbed output; however, after receiving perturbed outputs from many users, the data collector can make aggregate inferences pertaining to the general user population. In addition to receiving attention from the research community, LDP has also been deployed in products of large tech companies, including Google's RAPPOR for analyzing Chrome browser settings [Erlingsson et al., 2014], Apple iOS to collect popular emojis and trending words [app, 2020, Thakurta et al., 2017], and Microsoft Windows 10 for collecting application telemetry [Ding et al., 2017].

One domain that would highly benefit from adopting LDP is *geospatial data*, i.e., data relating to users' locations. Location data is inherently sensitive and collecting users' location data raises a multitude of privacy concerns [De Montjoye et al., 2013, Ma et al., 2013, Fiore et al., 2020]. Yet, geospatial statistics are useful for many purposes such as measurement of crowdedness for points-of-interest (shops, restaurants, museums), traffic density estimation, traffic navigation, passenger demand analysis, and so forth. Interest in geospatial density and mobility statistics have especially increased during the COVID-19 pandemic, with companies like Google[1],

[1]Google COVID-19 Community Mobility Reports – `https://www.google.com/covid19/mobility/`

Apple[2] and Uber[3] sharing mobility-related statistics to provide insights about the pandemic. Consequently, there is an active and growing need for novel methods to collect geospatial statistics while ensuring users' privacy using LDP.

Towards addressing this need, this thesis studies the problem of answering spatial density queries under LDP, which is a fundamental primitive for many downstream data analysis tasks. A spatial density query is a query of the form: *"How many users are located in the geographical area bounded by latitudes (X,Y) and longitudes (Z,T)?"* or *"How many users are traveling on road X?"*. Many of the aforementioned tasks such as measurement of crowdedness, estimation of traffic density, passenger demand analysis, etc. fundamentally rely on spatial density queries.

Our solution for answering spatial density queries under LDP consists of four main steps. The first step is *partitioning* a large query set and user population into smaller subsets, in order to reduce sensitivity and to benefit from LDP's principle of dividing users. We propose three basic partitioning strategies (*Singleton*, *Holistic* and *Random*), and analyze their advantages and disadvantages in terms of sampling error and noise error. Based on our analysis, we propose an improved partitioning strategy called *Advanced Partitioning*, whose superior accuracy is shown also experimentally. The second step is *finding the sensitivity* of a query set (or partition). For this purpose, we adapt graph-based modeling of query sets from the centralized differential privacy literature [Inan et al., 2016, Inan et al., 2020] to our setting. Our proposed *Advanced Partitioning* strategy also leverages this graph-based model by formulating the ideal partitioning problem as a vertex coloring problem, which constitutes another extension of the work in [Inan et al., 2016, Inan et al., 2020].

The third and fourth steps of our solution are *noisy response computation* on the user-side and *estimation of query answers* on the server-side. For these purposes, we initially adapt and use two LDP protocols: a binary version of Generalized Randomized Response and a modified version of Basic RAPPOR [Erlingsson et al., 2014, Wang et al., 2017]. However, in order to improve estimation accuracy, we

[2]Apple COVID-19 Mobility Trends Reports – `https://covid19.apple.com/mobility`

[3]Uber Movement – `https://movement.uber.com/?lang=en-US`

incorporate a third protocol into our solution by proposing an extension for one of the existing LDP protocols: Optimized Unary Encoding (OUE) [Wang et al., 2017]. Although OUE achieves highly accurate results in frequency estimation, the fact that it uses unary encoding is a limitation that prevents it from being directly employed in our solution. To overcome this limitation, we extend OUE to support arbitrary bitvector encodings, and call the resulting protocol Optimized Bitvector Encoding (OBE). We incorporate the extended user-side noisy response computation and server-side estimation procedures of OBE into our solution. Note that while OBE is a component of our solution, it is actually a generic LDP protocol that can be used independently in other LDP problems as well.

Combining the different partitioning strategies and LDP protocols, we obtain a total of 8 different approaches, all of which can be parsed as instances of our general four-step solution with different choices in the individual steps. We experimentally compare these 8 approaches using 4 real-world location datasets, two error metrics, varying sizes of query sets, and varying query characteristics. Experiment results demonstrate the superiority of our proposed approaches – in a large majority of the experiments, lowest error is obtained when our proposed *Advanced Partitioning* strategy is used together with the *OBE* protocol (OBE-ADV).

1.1 Contributions

In this thesis, we design and develop a solution for answering spatial density queries under LDP, consisting of four steps: partitioning, finding sensitivity, user-side noisy response computation, and server-side estimation. We consider four different strategies for partitioning, and three different LDP protocols for user-side noisy response computation and server-side estimation.

Additionally, for finding sensitivity, we adapt the technique of graph-based modeling of query sets [Inan et al., 2020] to our setting. We further extend this technique by proposing *Advanced Partitioning*, which formulates the partitioning problem as a vertex coloring problem on the graph model.

Furthermore, we propose a new protocol called OBE, which is an extension of an

existing protocol (OUE) in Chapter 4. In addition to reducing error in our solution, OBE can be used outside of our solution in other LDP problems as well.

Finally, we perform an extensive experimental evaluation of various combinations of partitioning strategies and LDP protocols using real-world datasets in Chapter 6. Results show the superiority of our proposed *Advanced Partitioning* strategy and *OBE* protocol.

1.2 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we present and discuss related work. In Chapter 3, we give preliminaries for the rest of the sections, such as the formal definitions of spatial density queries, LDP, and graph-based modeling of query sets. In Chapter 4, we describe the OBE protocol in detail, providing formal proofs for the privacy of its user-side perturbation and unbiasedness of its server-side estimation. In Chapter 5, we explain our solution for answering spatial density queries under LDP. We provide experimental analysis and comparison in Chapter 6. Chapter 7 concludes this thesis.

Chapter 2

RELATED WORK

2.1 Differential Privacy

Differential Privacy (DP) is the centralized privacy notion that precedes LDP where privacy protection mechanisms are applied at the database level [Dwork, 2006a]. Several works have studied the protection of geolocation datasets using DP. For answering count queries on two-dimensional geospatial datasets, [Qardaji et al., 2013] proposed a grid-based approach where the dataset is adaptively partitioned into several grids. [Ho and Ruan, 2011] studied the usage of differentially private region quadtrees for location databases that allows pattern mining. [Cormode et al., 2012] developed private versions of popular spatial indexing methods such as quadtrees, kd-trees and Hilbert R-trees. [Zhang et al., 2016] proposed PrivTree, a hierarchical method for releasing spatial distributions. A benchmark study by [Hay et al., 2016] provides a comparison of different DP query answering and histogram release algorithms. [Mir et al., 2013] and [Acs and Castelluccia, 2014] studied the release of spatio-temporal densities of urban areas using call data records.

Instead of spatio-temporal statistics, it is also possible to publish synthetic location trace (trajectory) datasets while satisfying DP. Then, desired statistics and/or query answers can be derived from the synthetic datasets. [Shao et al., 2013] publish trajectories using a weaker application of DP by injecting noise to locations. [Chen et al., 2012a] and [Chen et al., 2012b] publish sequential datasets using the prefix tree and ngram data models, respectively. [He et al., 2015] developed DPT for trajectory synthesis with the novelty of hierarchical reference systems. DP-Star and AdaTrace systems were developed in [Gursoy et al., 2018a] and [Gursoy et al., 2018b], which were empirically shown to provide higher utility compared to prior works. An optimized version of AdaTrace, called OptaTrace, was developed in [Gur-

soy et al., 2020].

Although the works surveyed in this section are at the intersection of DP and location data, they have a different privacy assumption (DP) compared to the privacy goal in this thesis (LDP), therefore they are not comparable to the solution developed in this thesis. An important distinction between DP and LDP is that DP requires all data to be located in a centralized database, whereas LDP allows users' data to remain on their personal devices. Consequently, LDP is more suitable and desirable for the setting considered in our work.

2.2 Local Differential Privacy

LDP has been receiving increasing attention from the industry and academia in recent years. In [app, 2020], Apple developed client-side and server-side LDP algorithms for collecting user data to determine several different statistics such as popular emojis, memory usage, word discovery and typing history. Mechanisms proposed in [Ding et al., 2017] are used by Microsoft to collect application telemetry data. Google also [Erlingsson et al., 2014] proposed RAPPOR protocol for collecting user statistics in Chrome. [Cormode et al., 2018a] analyzed large-scale adoptions of LDP in leading technology companies including Apple, Microsoft and Google.

One of the most fundamental tasks in LDP has been *frequency estimation*, and several protocols were developed towards this task. One of the first protocols under LDP is proposed in [Kairouz et al., 2014], which are called staircase mechanisms aimed for extremal privatization of collective personal data. [Bassily and Smith, 2015] studied efficient 1-bit protocols under the local setting that produce succinct histograms. [Wang et al., 2016] developed a high utility protocol specifically for statistical learning under local ϵ-LDP. Google's RAPPOR [Erlingsson et al., 2014] is one of the first LDP protocols that is used in mass data collection in commercial use for collecting anonymous usage statistics for Google Chrome. In [Wang et al., 2017] several optimized LDP protocols are introduced for private frequency estimation under different encoding mechanisms such as histogram encoding, unary encoding and local hashing. Among these protocols, we adapted and used the GRR and

RAPPOR protocols in our solution; in addition, we proposed an extension to the OUE protocol called OBE.

LDP has been used in several other domains and tasks that are orthogonal to the setting considered in this thesis, such as heavy hitter and frequent term discovery [Bassily et al., 2017, Wang et al., 2019c, Wang et al., 2018a, Fanti et al., 2016], frequent item and itemset mining from set-valued data [Qin et al., 2016, Wang et al., 2018b], key-value store analysis [Ye et al., 2021, Gu et al., 2020], mean and numerical distribution estimation [Duchi et al., 2018, Wang et al., 2019a, Li et al., 2020], marginal release from high-dimensional datasets [Cormode et al., 2018b, Ren et al., 2018, Zhang et al., 2018], and deep learning [Arachchige et al., 2019, Truex et al., 2020].

Work that is more closely related to this thesis' setting can be studied under two categories. First category is answering range and/or analytical queries under LDP. For answering 1-D range queries on a single attribute, [Cormode et al., 2019] proposed the use of Haar wavelet transforms under LDP. [Wang et al., 2019b] developed HIO for answering multi-dimensional analytical SQL-like queries on fact tables under LDP. [Yang et al., 2020] developed HDG for answering multi-attribute range queries on tabular data, leveraging the idea of combining low-dimensional grids to answer higher-dimensional queries. There have also been studies on answering linear and range queries with relaxations of LDP, such as metric-based LDP or local d-privacy [Xiang et al., 2020, Gu et al., 2019]. However, works in this category are not comparable to ours, since they are either not considering geolocation data (and the rich set of arbitrary-shaped density queries that may exist in geolocation space) or they assume privacy definitions that are different from LDP.

The second category is applications of LDP to geolocation data. Under this category, [Chen et al., 2016] studied spatial data aggregation through user clustering and user-side local randomizers. Their approach satisfies a privacy notion called (τ, ε)-PLDP, where each user's privacy specification is personalized by different spatial region τ and different privacy level ε. [Kim et al., 2018] applied LDP to collect indoor positioning data of users, and proposed a statistic-based method and an Expectation-Maximization method for estimation. [Zhao et al., 2020] provide a

targeted survey of DP/LDP literature with the goal of identifying future research directions at the intersection of Internet of Vehicles (IoV) and LDP; but this is a survey/vision paper without novel algorithms or protocols. The recent work of [Hong et al., 2021] proposes a *square mechanism* for collecting geospatial data under LDP while reducing the MSE between actual and collected locations, for improved personalized services. This is an orthogonal problem to ours since their solution aims to improve utility for each particular user, whereas our solution aims to maximize utility of aggregate statistics across the population.

In short, existing work under the first category (query answering under LDP) does not consider geolocation data or assume privacy definitions that are different than LDP. Existing work under the second category (applications of LDP to geolocation data) does not address answering spatial density queries. Our work in this thesis fills this gap.

Chapter 3

BACKGROUND

3.1 Problem Setting and Notation

Consider a two-dimensional geolocation space $\mathcal{L}$ and user population $\mathcal{P}$. For user $u \in \mathcal{P}$, at any given time, the user's true location $\ell_u \in \mathcal{L}$ is a tuple consisting of (latitude, longitude) coordinates. A spatial density query q is defined by a geographical area $A_q \subseteq \mathcal{L}$ and aims to estimate the number of users in that area.

We say that a user u answers query q positively if and only if $\ell_u \in A_q$. We denote this by an indicator function φ:

$$\varphi(u, q) = \begin{cases} 1 & \text{if } \ell_u \in A_q \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Then, the true answer of a spatial density query q is:

$$ans_q = \sum_{u \in \mathcal{P}} \varphi(u, q) \tag{3.2}$$

A collection of several spatial density queries is called a *query set*, denoted by: $\mathcal{Q} = \{q_1, q_2, q_3, ...\}$. We denote by $ans_{\mathcal{Q}}$ the answers to all queries in $\mathcal{Q}$, i.e.:

$$ans_{\mathcal{Q}} = \{ans_{q_1}, ans_{q_2}, ans_{q_3}, ...\} \tag{3.3}$$

The problem we study in this thesis can be stated as follows. Consider a server who holds a query set $\mathcal{Q}$ and wants to compute $ans_{\mathcal{Q}}$. Computing $ans_{\mathcal{Q}}$ may be desired for a number of reasons, e.g., learning the crowdedness of points-of-interest (shops, restaurants), providing traffic navigation service by measuring traffic density on roads of interest, adaptively improving service quality via resource allocation to dense regions, and so forth. However, computing $ans_{\mathcal{Q}}$ requires either collecting each user's location ℓ_u or asking each user to answer $\mathcal{Q}$ according to his/her true

ℓ_u, both of which violates the privacy of the users by exposing their true location. Thus, the goal of our work is to design and implement a solution such that $ans_{\mathcal{Q}}$ can be computed while the privacy each user's location remains protected via local differential privacy.

3.2 Local Differential Privacy

Local Differential Privacy (LDP) is a popular privacy notion that has been receiving increasing attention in recent years. It has also been deployed in consumer-facing products of companies such as Apple, Google and Microsoft [Ding et al., 2017, Erlingsson et al., 2014, app, 2020]. Its main difference compared to the previous, *centralized* version of differential privacy [Dwork, 2006b] is that it enables users' data to be perturbed *before* they are collected. That is, under LDP, each user locally perturbs their true data on their own device using a randomized algorithm Ψ, and the perturbed output is sent to the server. Since only the perturbed outputs are observed by the server and not the users' true data, LDP can be used in scenarios where the users do not trust the server. LDP fits our problem setting nicely: While the query set $\mathcal{Q}$ is being answered, users will not have to share their true locations or true query responses with the server; only their perturbed responses will be observed by the server. Hence, their privacy remains protected by LDP.

Definition 1 (ε-LDP)**.** A randomized algorithm Ψ satisfies ε-local differential privacy (ε-LDP), where $\varepsilon > 0$, if and only if for any two inputs v_1, v_2 in universe $\mathcal{U}$, it holds that:

$$\forall y \in Range(\Psi): \quad \frac{\Pr[\Psi(v_1) = y]}{\Pr[\Psi(v_2) = y]} \leq e^{\varepsilon} \tag{3.4}$$

where $Range(\Psi)$ denotes the set of all possible outputs of Ψ.

Intuitively, ε-LDP ensures that given the perturbed output y, it is not possible for the server or for an adversary to distinguish whether the original true value was v_1 or v_2 beyond the ratio e^{ε}. Here, ε controls the privacy level. Lower ε yields stronger privacy.

The concept of *sensitivity* will be important in determining the amount of perturbation necessary to achieve LDP. Sensitivity measures the maximum number of queries in a query set $\mathcal{Q}$ that could be simultaneously impacted by one user, i.e., how many queries a single user u can simultaneously answer positively, considering the user is located in a particular location ℓ_u. Intuitively, if sensitivity is high, higher magnitude of perturbation is necessary to achieve LDP.

Definition 2 (Sensitivity)**.** Let $\mathcal{Q}$ denote a set of queries. We say that the sensitivity of $\mathcal{Q}$, denoted $\Delta_{\mathcal{Q}}$, is the maximum number of queries q that could be answered positively by any one plausible user u with location ℓ_u:

$$\Delta_{\mathcal{Q}} = \max_{\ell_u \in \mathcal{L}} \Big(\sum_{q \in \mathcal{Q}} \varphi(u, q) \Big) \tag{3.5}$$

When clear in the context, we drop the subscript in $\Delta_{\mathcal{Q}}$ and simply write Δ.

Principle of Dividing Users: In LDP, several previous works [Wang et al., 2017, Yang et al., 2020, Zhang et al., 2018, Cormode et al., 2019] point out that when multiple pieces of information are to be collected from a user population, it is better to divide (partition) users into multiple groups and collect different pieces of information from each group. This way, each user spends their full ε budget on one piece of information. In contrast, dividing the privacy budget ε into several pieces via sequential composition was shown to yield lower accuracy. Thus, our solution takes advantage of the principle of dividing users.

3.3 Graph-Based Modeling of Query Sets

Modeling query sets via graphs was proposed in [Inan et al., 2020] and used in [Esmerdag et al., 2016, Inan et al., 2016, Gursoy et al., 2017] under centralized differential privacy (DP). Modeling a query set as a graph enables efficient computation of its sensitivity via graph algorithms (such as maximum clique problem), and by knowing the sensitivity of a query set, appropriate amount of noise can be added to query answers so that DP is satisfied. In this thesis, we adapt some of these findings to the context of LDP and spatial queries. Furthermore, we extend

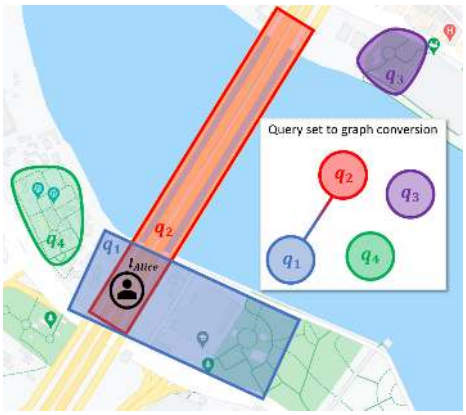

Figure 3.1: Query set to graph conversion

the state-of-the-art by showing how *vertex coloring*, another graph algorithm, can be applied to divide a large query set into smaller subsets, each with $\Delta = 1$.

Definition 3 (Graph model of $\mathcal{Q}$)**.** Let $\mathcal{Q} = \{q_1, q_2, q_3, ...\}$ be a query set. The graph model of $\mathcal{Q}$ is an undirected graph $G = (V, E)$ with vertex set V and edge set E, such that:

- Each query $q \in \mathcal{Q}$ is represented as a vertex in V.
- There exists an edge between two vertices if and only if the geographical regions of the corresponding two queries intersect with each other, i.e.:

$$E = \{(q_i, q_j) \mid A_{q_i} \cap A_{q_j} \neq \emptyset, \quad \forall (q_i, q_j) \in \mathcal{Q} \times \mathcal{Q}\}$$

An example conversion of a query set $\mathcal{Q}$ to its graph model $G(V, E)$ is shown in Figure 3.1. The query set consists of four queries: $\mathcal{Q} = \{q_1, q_2, q_3, q_4\}$, e.g., q_1 estimates density in a neighborhood, q_2 estimates traffic on the bridge, q_3 and q_4 estimate number of visitors in the museum and the park, respectively. For user Alice with location ℓ_{Alice}, $\varphi(Alice, q_1) = 1$ and $\varphi(Alice, q_3) = 0$. The graph model of $\mathcal{Q}$ is shown in Figure 3.1. In the graph, there is only one edge (between q_1 and q_2) since q_1 and q_2 are the only two pairwise intersecting queries. For this $\mathcal{Q}$, according to Definition 2, we have: $\Delta_{\mathcal{Q}} = 2$.

Chapter 4

OPTIMIZED BITVECTOR ENCODING (OBE)

Before describing our solution for answering spatial density queries under LDP, we propose an extension to an existing LDP protocol called Optimized Unary Encoding (OUE). The extended protocol, called Optimized Bitvector Encoding (OBE), is used in our spatial query answering solution.

OUE is an LDP protocol that was proposed in [Wang et al., 2017]. It uses unary encoding in which user u's true response is encoded into a *unary* bitvector B_u such that only one position in B_u contains a 1 bit and remaining positions contain 0 bits. The OUE protocol is suitable for the frequency estimation task which was the main goal of [Wang et al., 2017], and it was shown to provide highly accurate results. However, since OUE only supports unary encoding, it is not suitable for LDP tasks which require more general bitvector encoding with potentially many 1s in B_u, such as our query answering task in which a user's true answers could be encoded into a bitvector such that $B_u[i] = 1$ iff $\varphi(u, q_i) = 1$, and there exist multiple queries $q_i \in \mathcal{Q}$ for which $\varphi(u, q_i) = 1$. To address this shortcoming, we extend OUE such that arbitrary number of 1s can be supported in B_u. We denote the maximum number of 1s that can exist in B_u by Δ, since Δ corresponds to the sensitivity of $\mathcal{Q}$, i.e., maximum number of positive bit responses that may exist in B_u. We call the extended protocol Optimized Bitvector Encoding (OBE).

Note that similar to many other LDP protocols, our OBE protocol can be used in arbitrary LDP tasks – while we leverage it in solving the spatial query answering problem, the protocol itself is generic and it is not limited to this task.

4.1 Existing OUE Protocol

First, we give the technical details of the existing version of the OUE protocol from [Wang et al., 2017]. As in many other LDP protocols, OUE consists of two main components: (i) user-side perturbation to satisfy LDP on users' devices, and (ii) server-side estimation after collecting perturbed data from the user population.

User-Side Perturbation in OUE. Let B_u denote user u's bitvector. Recall that B_u is unary, i.e., only one position in B_u contains a 1 bit and all remaining positions are 0. The perturbation mechanism Ψ of OUE takes B_u as input and outputs perturbed bitvector B'_u as follows:

$$\Pr\Big[B'_u[i] = \Psi(B_u[i]) = 1\Big] = \begin{cases} \frac{1}{2} & \text{if } B_u[i] = 1 \\ \frac{1}{e^{\varepsilon}+1} & \text{if } B_u[i] = 0 \end{cases} \tag{4.1}$$

That is, each position $i \in [1, |B_u|]$ is considered one-by-one, and the existing bit is either kept or flipped with different probabilities. The perturbation probabilities are chosen in a way that minimizes the variance of server-side estimation, so that the protocol's accuracy remains high when $|B|$ is large [Wang et al., 2017]. The perturbed bitvector B'_u is sent to the server.

Server-Side Estimation in OUE. The server collects perturbed bitvectors B'_u from users $u \in \mathcal{P}$. Then, the server computes the reported count of position i, denoted $\widehat{C}(i)$, as:

$$\widehat{C}(i) = \sum_{u \in \mathcal{P}} B'_u[i] \tag{4.2}$$

The server's estimate of how any users have $B_u[i] = 1$ in their original bitvectors, denoted $\bar{C}(i)$, is computed as:

$$\bar{C}(i) = \frac{2 \cdot \big((e^{\varepsilon}+1) \cdot \widehat{C}(i) - |\mathcal{P}|\big)}{e^{\varepsilon}-1} \tag{4.3}$$

By computing $\bar{C}(i)$ independently for each $i \in [1, |B_u|]$, the server is able to estimate the frequency of each index.

4.2 User-side Perturbation in OBE

Our extension of OUE to OBE requires modification of both the user-side perturbation and server-side estimation. In this section, we describe the user-side perturbation in OBE and prove that it satisfies ε-LDP. In the next section, we describe the server-side estimation in OBE and prove that it provides an unbiased estimate.

Let B_u denote user u's bitvector. Different from OUE, here, we allow arbitrary number of 1s in B_u, where Δ is the maximum number of 1s allowed. The perturbation mechanism Ψ of OBE takes B_u as input and outputs perturbed bitvector B'_u as follows:

$$\Pr\Big[B'_u[i] = \Psi(B_u[i]) = 1\Big] = \begin{cases} \frac{1}{2} & \text{if } B_u[i] = 1 \\ \frac{1}{e^{\varepsilon/\Delta}+1} & \text{if } B_u[i] = 0 \end{cases} \tag{4.4}$$

User u sends perturbed bitvector B'_u to the server.

Theorem 1. User-side perturbation of OBE satisfies ε-LDP.

Proof. Let B_{u_1}, B_{u_2} denote two input bitvectors to be perturbed, and let B' denote a perturbed bitvector (output produced by OBE). Since Ψ of OBE does not change the length of the bitvectors, we have $|B_{u_1}| = |B_{u_2}| = |B'|$. By definition of LDP, we need to show that:

$$\forall B' \in Range(\Psi): \quad \frac{\Pr[\Psi(B_{u_1}) = B']}{\Pr[\Psi(B_{u_2}) = B']} \leq e^{\varepsilon} \tag{4.5}$$

We define two probabilities P_{max}, P_{min}:

$$P_{max} = \max\Big(\Pr[\Psi(B_{u_1}) = B']\Big)$$

$$P_{min} = \min\Big(\Pr[\Psi(B_{u_2}) = B']\Big)$$

Hence, if $P_{max}/P_{min} \leq e^{\varepsilon}$, then Equation 4.5 is satisfied. Next, notice that P_{max} is achieved when all bits of B_{u_1} are the same as the bits of B', i.e. $B_{u_1}[i] = B'[i], \ \forall i \in [1, |B'|]$. Similarly, P_{min} is achieved when all bits of B_{u_2} are different from the bits of B', i.e. $B_{u_2}[i] \neq B'[i], \ \forall i \in [1, |B'|]$.

Let n denote the number of 1 bits in B'. In the remainder of the proof, we consider 4 different cases depending on: (i) whether $n \leq \Delta$ or not, (ii) whether

$|B'| \geq n + \Delta$ or not. We compute ${}^{P_{max}}/_{P_{min}}$ under each of the 4 cases and show that ${}^{P_{max}}/_{P_{min}} \leq e^{\varepsilon}$ holds under each case.

Case 1. Assume $n \leq \Delta$ and $|B'| \geq n + \Delta$. We have:

$$P_{max} = \left(\frac{1}{2}\right)^{n} \left(\frac{e^{\varepsilon/\Delta}}{e^{\varepsilon/\Delta}+1}\right)^{|B'|-n}$$

$$P_{min} = \left(\frac{1}{e^{\varepsilon/\Delta}+1}\right)^{n} \left(\frac{1}{2}\right)^{\Delta} \left(\frac{e^{\varepsilon/\Delta}}{e^{\varepsilon/\Delta}+1}\right)^{|B'|-n-\Delta}$$

Consequently,

$$\frac{P_{max}}{P_{min}} = e^{\varepsilon} \cdot \left(\frac{e^{\varepsilon/\Delta}+1}{2}\right)^{n-\Delta} \leq e^{\varepsilon} \tag{4.6}$$

The last step follows from the fact that, for all possible values of ε and Δ, ${}^{\varepsilon}/_{\Delta} > 0$ and thus $e^{\varepsilon/\Delta} > 1$. The quantity $\frac{e^{\varepsilon/\Delta}+1}{2}$ is therefore greater than 1, but since $n \leq \Delta$, it is raised to a non-positive power, leading to the final result.

Case 2. Assume $n \leq \Delta$ and $|B'| \leq n + \Delta$. We have:

$$P_{max} = \left(\frac{1}{2}\right)^{n} \left(\frac{e^{\varepsilon/\Delta}}{e^{\varepsilon/\Delta}+1}\right)^{|B'|-n}$$

$$P_{min} = \left(\frac{1}{e^{\varepsilon/\Delta}+1}\right)^{n} \left(\frac{1}{2}\right)^{|B'|-n}$$

Consequently,

$$\frac{P_{max}}{P_{min}} = \left(\frac{1}{2}\right)^{2n-|B'|} (e^{\varepsilon/\Delta})^{|B'|-n} \left(\frac{1}{e^{\varepsilon/\Delta}+1}\right)^{|B'|-2n} \tag{4.7}$$

We observe that Equation 4.7 is monotonically increasing with respect to $|B'|$. This is because after removing all multipliers in Equation 4.7 that do not depend on $|B'|$, we obtain:

$$\left(\frac{2e^{\varepsilon/\Delta}}{e^{\varepsilon/\Delta}+1}\right)^{|B'|}$$

For all possible values of ε and Δ, which satisfy $\varepsilon \geq 0$ and $\Delta > 0$, it holds that: $2e^{\varepsilon/\Delta} \geq e^{\varepsilon/\Delta} + 1$. Thus, the fraction is > 1, and the fraction raised to $|B'|$ is monotonically increasing with respect to $|B'|$.

In order to compute the maximum value of Equation 4.7, we must therefore plug $|B'| = n + \Delta$ into it, since this is the maximum value $|B'|$ can take under Case 2. Then:

$$\frac{P_{max}}{P_{min}} = e^{\varepsilon} \cdot \left(\frac{e^{\varepsilon/\Delta}+1}{2}\right)^{n-\Delta} \leq e^{\varepsilon} \tag{4.8}$$

Case 3. Assume $n \geq \Delta$ and $|B'| \geq n + \Delta$. We have:

$$P_{max} = \left(\frac{1}{2}\right)^{\Delta} \left(\frac{1}{e^{\varepsilon/\Delta}+1}\right)^{n-\Delta} \left(\frac{e^{\varepsilon/\Delta}}{e^{\varepsilon/\Delta}+1}\right)^{|B'|-n}$$

$$P_{min} = \left(\frac{1}{2}\right)^{\Delta} \left(\frac{1}{e^{\varepsilon/\Delta}+1}\right)^{n} \left(\frac{e^{\varepsilon/\Delta}}{e^{\varepsilon/\Delta}+1}\right)^{|B'|-n-\Delta}$$

Consequently,

$$\frac{P_{max}}{P_{min}} = \frac{\left(\frac{e^{\varepsilon/\Delta}}{e^{\varepsilon/\Delta}+1}\right)^{\Delta}}{\left(\frac{1}{e^{\varepsilon/\Delta}+1}\right)^{\Delta}} = e^{\varepsilon} \tag{4.9}$$

Case 4. Assume $n \geq \Delta$ and $|B'| \leq n + \Delta$. We have:

$$P_{max} = \left(\frac{1}{2}\right)^{\Delta} \left(\frac{1}{e^{\varepsilon/\Delta}+1}\right)^{n-\Delta} \left(\frac{e^{\varepsilon/\Delta}}{e^{\varepsilon/\Delta}+1}\right)^{|B'|-n}$$

$$P_{min} = \left(\frac{1}{2}\right)^{|B'|-n} \left(\frac{1}{e^{\varepsilon/\Delta}+1}\right)^{n}$$

Consequently,

$$\frac{P_{max}}{P_{min}} = \left(\frac{1}{2}\right)^{n+\Delta-|B'|} \left(\frac{1}{e^{\varepsilon/\Delta}+1}\right)^{|B'|-n-\Delta} (e^{\varepsilon/\Delta})^{|B'|-n} \tag{4.10}$$

We observe that Equation 4.10 is monotonically increasing with respect to $|B'|$. The derivation of this is identical to that under Case 2, therefore we omit it here. In order to compute the maximum possible value of Equation 4.10, we plug $|B'| = n + \Delta$ into it, which is the maximum value $|B'|$ can take under Case 4. Then, we obtain:

$$\frac{P_{max}}{P_{min}} = (e^{\varepsilon/\Delta})^{\Delta} = e^{\varepsilon} \tag{4.11}$$

The proof is complete, since we showed in Equations 4.6, 4.8, 4.9, 4.11 that under each of the 4 cases, $P_{max}/P_{min} \leq e^{\varepsilon}$ holds. □

4.3 Server-side Estimation in OBE

After the server receives perturbed bitvectors B'_u from users $u \in \mathcal{P}$, in order to recover how many users' true bitvectors B_u contain $B_u[i] = 1$, the server needs to perform estimation $\forall i \in [1, |B_u|]$. First, the reported count of position i is computed:

$$\widehat{C}(i) = \sum_{u \in \mathcal{P}} B'_u[i] \tag{4.12}$$

Then, the server computes the estimate $\bar{C}(i)$ as:

$$\bar{C}(i) = \frac{2 \cdot ((e^{\varepsilon/\Delta} + 1) \cdot \widehat{C}(i) - |\mathcal{P}|)}{e^{\varepsilon/\Delta} - 1} \tag{4.13}$$

A desirable property of estimation is being *unbiased*. That is, in expectation, the estimate $\bar{C}(i)$ is equal to the true number of users whose bitvectors contain $B_u[i] = 1$. Theorem 2 proves that the above estimation for OBE is indeed unbiased.

Theorem 2. Let γ denote the fraction of users in $\mathcal{P}$ whose true bitvectors contain $B_u[i] = 1$. Then, $\bar{C}(i)$ is an unbiased estimator of $|\mathcal{P}| \cdot \gamma$.

Proof. We need to show: $\mathbb{E}\left[\bar{C}[i]\right] = |\mathcal{P}| \cdot \gamma$. First, observe from Equation 4.13 that:

$$\mathbb{E}\left[\bar{C}[i]\right] = \frac{2 \cdot ((e^{\varepsilon/\Delta} + 1) \cdot \mathbb{E}\left[\widehat{C}(i)\right] - |\mathcal{P}|)}{e^{\varepsilon/\Delta} - 1} \tag{4.14}$$

We can compute $\mathbb{E}\left[\widehat{C}(i)\right]$ using the perturbation probabilities defined in Equation 4.4:

$$\mathbb{E}\left[\widehat{C}[i]\right] = \frac{1}{2} \cdot |\mathcal{P}| \cdot \gamma + \frac{1}{e^{\varepsilon/\Delta} + 1} \cdot |\mathcal{P}| \cdot (1 - \gamma) \tag{4.15}$$

Plugging $\mathbb{E}\left[\widehat{C}[i]\right]$ from Equation 4.15 into Equation 4.14, we obtain:

$$\begin{aligned}
\mathbb{E}\left[\bar{C}[i]\right] &= \frac{2|\mathcal{P}| \cdot (e^{\varepsilon/\Delta} + 1) \cdot \left[\frac{\gamma}{2} + \frac{1-\gamma}{e^{\varepsilon/\Delta}+1}\right] - 2|\mathcal{P}|}{e^{\varepsilon/\Delta} - 1} \\
&= \frac{|\mathcal{P}| \cdot \gamma \cdot (e^{\varepsilon/\Delta} + 1) + 2|\mathcal{P}| \cdot (1 - \gamma) - 2|\mathcal{P}|}{e^{\varepsilon/\Delta} - 1} \\
&= \frac{|\mathcal{P}| \cdot \gamma \cdot (e^{\varepsilon/\Delta} + 1) - 2|\mathcal{P}| \cdot \gamma}{e^{\varepsilon/\Delta} - 1} \\
&= \frac{|\mathcal{P}| \cdot \gamma \cdot (e^{\varepsilon/\Delta} - 1)}{e^{\varepsilon/\Delta} - 1} \\
&= |\mathcal{P}| \cdot \gamma
\end{aligned} \tag{4.16}$$

□

Chapter 5

ANSWERING SPATIAL QUERIES UNDER LDP

In this chapter, we describe our solution for answering spatial density queries under LDP. We start with a general overview of our solution in Section 5.1, and then explain each step in more detail in the next sections.

5.1 Solution Overview

The overview of our solution is provided in Algorithm 1. Given query set $\mathcal{Q}$, user population $\mathcal{P}$ and privacy level ε, our solution computes the answers to the queries in $\mathcal{Q}$ while satisfying ε-LDP for each user. The algorithm consists of four main steps. First, on the server-side, $\mathcal{Q}$ and $\mathcal{P}$ are PARTITIONed into n pairs of subsets. There are multiple partitioning strategies; these are presented in Sections 5.2 and 5.3. All of the partitioning strategies satisfy $\mathcal{Q} = \cup_{i=1}^{n} Q_i$ and $\mathcal{P} = \cup_{i=1}^{n} P_i$. The result of partitioning is n pairs of (Q_i, P_i) such that user subset P_i is responsible for answering the query subset Q_i. Then, for each (Q_i, P_i) pair, the server sends Q_i to users $u \in P_i$ and awaits their responses. Each user locally computes a noisy LDP response on his/her device and sends it to the server in steps 2 and 3. The sensitivity of Q_i is computed by the FIND_SENSITIVITY function, and the perturbed LDP response is computed by the COMPUTE_RESPONSE function, taking into account sensitivity, Q_i and ε. The details of FIND_SENSITIVITY are presented in Section 5.4 and the details of COMPUTE_RESPONSE are presented in Section 5.5. After the server collects perturbed responses from the users, it estimates ans_{Q_i} using the ESTIMATE function. The details of estimation are presented in Section 5.6. Finally, on line 13 of Algorithm 1, the estimated answers ans_{Q_i} of each partition Q_i are combined to obtain $ans_{\mathcal{Q}}$.

Algorithm 1: Overview of our solution

Input : Query set $\mathcal{Q}$, user population $\mathcal{P}$, privacy budget ε

Output: Query answers $ans_{\mathcal{Q}}$

$(Q_1, P_1), (Q_2, P_2), .., (Q_n, P_n) \leftarrow$ PARTITION$(\mathcal{Q}, \mathcal{P})$
foreach (Q_i, P_i) **do**
  Server sends Q_i to users $u \in P_i$
  $responses \leftarrow \{\}$
  `/* Each user computes noisy LDP response and sends to server */`
  foreach $u \in P_i$ **do**
    $\Delta_{Q_i} \leftarrow$ FIND_SENSITIVITY(Q_i)
    $resp \leftarrow$ COMPUTE_RESPONSE$(Q_i, \varepsilon, \Delta_{Q_i})$
    User u sends $resp$ to server
    Server adds $resp$ to $responses$
  end
  `/* Server-side estimation from users' noisy responses */`
  $ans_{Q_i} \leftarrow$ ESTIMATE$(responses)$
end
$ans_{\mathcal{Q}} \leftarrow \bigcup_{i=1}^{n} ans_{Q_i}$
return $ans_{\mathcal{Q}}$

5.2 Basic Partitioning Strategies

The aim of the partitioning step is to divide $\mathcal{Q}$ and $\mathcal{P}$ into several (Q_i, P_i) pairs to determine which subset of the population P_i will answer which query subset Q_i. The rationale behind partitioning is to reduce Δ (thereby reducing the perturbation necessary to satisfy LDP) while leveraging LDP's principle of dividing users (recall from Section 3.2). To that end, we present 3 basic partitioning strategies: *Singleton Partitioning*, *Holistic Partitioning*, *Random Partitioning*. The advantages and disadvantages of these basic strategies inspire our *Advanced Partitioning* strategy.

Singleton Partitioning: The number of partitions n is set to $n = |\mathcal{Q}|$ and each query in $\mathcal{Q}$ becomes a partition Q_i on its own. That is, for $\mathcal{Q} = \{q_1, q_2, ..., q_n\}$, there are n partitions such that $Q_i = \{q_i\}$. Population $\mathcal{P}$ is divided into n mutually disjoint groups, each containing $|P_i| = \frac{|\mathcal{P}|}{|\mathcal{Q}|}$ users. Hence, under singleton partitioning, each user answers exactly one query, and each query is answered by $\frac{|\mathcal{P}|}{|\mathcal{Q}|}$ users.

Holistic Partitioning: The number of partitions n is set to $n = 1$ and the whole query set $\mathcal{Q}$ is treated as one partition. The result of holistic partitioning is a single partition $(P_1, Q_1) = (\mathcal{P}, \mathcal{Q})$. Hence, under holistic partitioning, each user answers the whole query set $\mathcal{Q}$, and each query is answered by $\mathcal{P}$ users.

Random Partitioning: As opposed to $n = |\mathcal{Q}|$ in singleton partitioning and $n = 1$ in holistic partitioning, under random partitioning n is a variable parameter: $1 < n < |\mathcal{Q}|$. $\mathcal{Q}$ is randomly divided into n query subsets denoted $Q_1, Q_2, ..., Q_n$, each containing $|Q_i| = \mathcal{Q}/n$ queries. Population $\mathcal{P}$ is also divided into n mutually disjoint groups denoted $P_1, P_2, ..., P_n$. Pairs (Q_i, P_i) are matched and provided as the partitioning result. Hence, under random partitioning, each user answers $|\mathcal{Q}|/n$ queries, and each query is answered by $|\mathcal{P}|/n$ users, where $1 < n < |\mathcal{Q}|$.

Analysis and Discussion: When answering spatial density queries with LDP, there will naturally be some error in $ans_{\mathcal{Q}}$ due to perturbation performed to satisfy LDP. In particular, we observe that there are two main sources of error under our framework:

- **Sampling error:** Consider a query $q \in \mathcal{Q}$. Since $\mathcal{P}$ is partitioned, q is typically not answered by the whole population $\mathcal{P}$, but rather a subset of the population P_i. Then, its answer must be generalized to $\mathcal{P}$ by multiplying it with $\frac{|\mathcal{P}|}{|P_i|}$. This introduces sampling error since the sample P_i may not perfectly represent the whole population $\mathcal{P}$, and any imperfection is aggravated by the $\frac{|\mathcal{P}|}{|P_i|}$ multiplication.

- **Noise error:** At first sight, one can propose to minimize sampling error by enforcing many users to answer each query q. However, for each $u \in \mathcal{P}$, this increases the number of queries u needs to answer, which causes increased sensitivity Δ. Unfortunately, increased Δ yields higher noise error due to LDP

perturbation, see for example Equation 4.4. Higher the LDP perturbation, higher the noise error in $ans_{\mathcal{Q}}$.

As explained above, sampling error and noise error have a natural trade-off between them. Indeed, among our 3 basic partitioning strategies, singleton partitioning minimizes noise error (since $|Q_i|$=1, it is guaranteed that Δ=1) but incurs heaviest sampling error. Holistic partitioning minimizes sampling error since each query is answered by whole $\mathcal{P}$, but incurs heaviest noise error. Random partitioning provides a trade-off between these two ends: by increasing n one gets closer to singleton partitioning and by decreasing n one gets closer to holistic partitioning.

After experimentally analyzing the 3 basic partitioning strategies, we observed that noise error is typically much more dominant than sampling error. That is, for the goal of minimizing **total** error in $ans_{\mathcal{Q}}$, we should first and foremost minimize noise error. This is achieved by enforcing Δ=1. Second, we observed with random partitioning that some randomly built query subsets Q_i can have Δ_{Q_i}=1 and $|Q_i| > 1$. Such Q_i achieve lowest noise error possible while simultaneously incurring lower sampling error compared to singleton partitioning since the queries are being answered by more users: $|\mathcal{P}|/n$ users rather than $|\mathcal{P}|/|\mathcal{Q}|$, where $n < |\mathcal{Q}|$.

5.3 Advanced Partitioning

The above observations motivate our *Advanced Partitioning* approach. Given $\mathcal{Q}$, the goal of advanced partitioning is to systematically find query subsets $Q_1, ..., Q_n$ such that: (i) each Q_i has Δ_{Q_i}=1 so that noise error is minimized, and (ii) n is kept as small as possible so that each query is answered by more users ($|\mathcal{P}|/n$), thereby sampling error is reduced simultaneously.

In order to solve this problem, advanced partitioning makes use of the graph modeling of query sets explained in Section 3.3. First, query set $\mathcal{Q}$ is converted to its graph model $G = (V, E)$. Then, observe that if two vertices q_i, q_j are not adjacent in G, their geographical regions A_{q_i}, A_{q_j} do not intersect. Hence, if we construct a query subset Q_k such that all queries in Q_k are pairwise not adjacent, then Δ_{Q_k} is 1. As a result, to achieve the first goal of advanced partitioning, we should partition $\mathcal{Q}$

into $Q_1, ..., Q_n$ such that each Q_k consists of queries that are not adjacent. In graph theory, this is known as *proper vertex coloring*.

Definition 4 (Proper Vertex Coloring)**.** Given an undirected graph G, assign a color to each vertex of G such that two adjacent vertices cannot have the same color. A coloring of G using at most n colors is called an n-coloring. The smallest number of colors needed to color a graph is called its *chromatic number*.

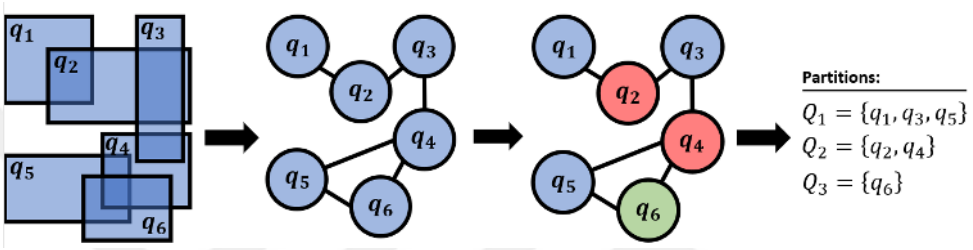

Figure 5.1: Sample execution of Advanced Partitioning with vertex coloring

The two goals of advanced partitioning stated at the beginning of this section can therefore be equivalently stated as: finding an n-coloring of G where n is G's chromatic number.

Unfortunately, vertex coloring is computationally hard. It is NP-complete to decide if a graph allows a k-coloring (except $k = 1$ or 2) and it is NP-hard to even compute the chromatic number of a graph [Garey et al., 1974, Johnson, 1985, Leven and Galil, 1983]. Therefore, we utilize a *greedy coloring* algorithm. Greedy coloring arranges vertices V and colors $\mathcal{C}$ as two separate ordered lists. Then, it iterates over the list of vertices, and for each vertex, it assigns to that vertex the first color that is available, i.e., the color which is at the lowest index of $\mathcal{C}$ but not already used by one of the neighbors of the vertex. The ordering of $\mathcal{C}$ is arbitrary and has no impact, but the ordering of V is important. We use the Largest-First method to order V, which is an effective strategy that was first put forward by Welsh and Powell [Welsh and Powell, 1967]. The Largest-First method utilizes the observation that vertices of low degree allow for a more flexible choice of color, whereas vertices with high degree are more difficult to color. Thus, it orders V in non-increasing order of degree, so that

Algorithm 2: Advanced Partitioning

Input : Query set $\mathcal{Q}$, user population $\mathcal{P}$

Output: Partitions $(Q_1, P_1), (Q_2, P_2), .., (Q_n, P_n)$

```
G ← GRAPH_MODEL(Q)
GREEDY_VERTEX_COLORING(G)
n ← number of colors in G
P_1, P_2, ..., P_n ← uniformly divide P into n mutually disjoint subsets
partitions ← {}
for color c_i where i ∈ [1, n] do
    Q_i ← vertices in G that were colored c_i
    partitions.add(Q_i,P_i)
end
return partitions
```

the more difficult vertices are colored first. Overall, the computational complexity of greedy coloring with the Largest-First method is linear time $O(|V| + |E|)$.

Putting everything together, advanced partitioning works as shown in Algorithm 2. First, $\mathcal{Q}$ is converted to its graph model G using the procedure in Section 3.3. Then, the greedy vertex coloring algorithm is applied to G which assigns a color to each vertex in G. Let n be the total number of distinct colors in the graph after all vertices have been colored. For each color c_i, a query subset Q_i is created which includes all vertices in G that were assigned color c_i. This yields $Q_1, Q_2, ..., Q_n$ query subsets. User population $\mathcal{P}$ is uniformly randomly divided into n mutually disjoint subsets $P_1, P_2, ..., P_n$. Collectively, advanced partitioning returns $(Q_1, P_1), (Q_2, P_2), ..., (Q_n, P_n)$. The theorem below proves that advanced partitioning achieves its design goal, i.e., $\Delta_{Q_i} = 1$ for each query subset Q_i.

Theorem 3. Consider partitioning of $\mathcal{Q}$ into $(Q_1, Q_2, ..., Q_n)$ by Algorithm 2. Then, the following properties hold:

1. No query is dropped, i.e., $\mathcal{Q} = Q_1 \cup Q_2 \cup ... \cup Q_n$.

2. Each query appears in at most one partition, i.e., for Q_i, Q_j such that $i \neq j$, $Q_i \cap Q_j = \emptyset$.

3. For each Q_i, we have: $\Delta_{Q_i} = 1$.

Proof. Vertex coloring assigns a color to every vertex and no vertex is left uncolored, thus Property 1 holds. Also, vertex coloring assigns exactly one color to each vertex and a single vertex cannot have multiple colors, thus Property 2 holds.

We prove Property 3 by contradiction. Suppose that there is a violation of Property 3, i.e., $\Delta_{Q_i} \geq 2$ for some Q_i. Algorithm 2 constructs one Q_i per different color; let c_i be the color corresponding to Q_i. For $\Delta_{Q_i} \geq 2$ to hold, there must be two queries $q_a, q_b \in Q_i$ which have intersecting regions A_{q_a},A_{q_b}. In that case, q_a and q_b are adjacent in G due to the graph modeling procedure from Section 3.3. However, by Definition 4, two adjacent vertices cannot have the same color in vertex coloring, thus their colors must be different, in which case they cannot both be in the same Q_i. This is a contradiction. □

A sample execution of Algorithm 2 is illustrated in Figure 5.1. After the graph model is obtained, one can observe that 3 colors are needed for vertex coloring (red, green, blue). Since there are 3 colors, we obtain 3 partitions: Q_1 consists of blue-colored vertices, Q_2 consists of red-colored vertices, Q_3 consists of green-colored vertices. One can also verify that Δ_{Q_i} is 1 for each partition, e.g., for Q_1, queries q_1, q_3, q_5 have no geographic intersection.

5.4 Finding Sensitivity

Before Q_i is answered by a user, its sensitivity Δ_{Q_i} must be computed by FIND_SENSITIVITY since Δ_{Q_i} is used in determining the perturbation amount in LDP protocols. While singleton partitioning and advanced partitioning specifically satisfy $\Delta_{Q_i} = 1$, our solution should support finding Δ_{Q_i} for arbitrary Q_i constructed by any partitioning strategy.

Fortunately, the graph model of Q_i provides the ability to find Δ_{Q_i} efficiently. In particular, Lemma 1 shows that subsets of queries that have common intersections

in the location space $\mathcal{L}$ form a *clique* in the graph model. A clique ω is a subset of vertices $\omega \subseteq V$ such that every two vertex in ω are adjacent, i.e., for $\forall a, b \in \omega$, $(a, b) \in E$. Using Lemma 1, we can derive Theorem 4 in which Δ_{Q_i} can be upper bounded by the *maximum clique size (MCS)* of the graph, which is equal to the number of vertices in the largest clique of the graph. Thus, by computing the MCS of the graph model of Q_i, we can find Δ_{Q_i}.

Lemma 1. Consider query set Q_i and its graph model $G(V, E)$. Let $Q_i^{sub} \subseteq Q_i$ be a query subset such that all queries in Q_i^{sub} have a joint common intersection, i.e.:

$$\Big(\bigcap_{q \in Q_i^{sub}} A_q \Big) \neq \emptyset \tag{5.1}$$

Then, the vertices representing Q_i^{sub} form a clique in G, with clique size $|Q_i^{sub}|$.

Proof. For a common intersection to exist and therefore Equation 5.1 to hold, for all pairs $q_j, q_k \in Q_i^{sub}$, it must hold that $A_{q_j} \cap A_{q_k} \neq \emptyset$. In that case, according to the graph constructed by Def. 3, there exists an edge between all pairs q_j and q_k, i.e., $\forall q_j, q_k \in Q_i^{sub}$, $(q_j, q_k) \in E$. By definition of a clique, this corresponds to a clique in G, which contains the vertices of Q_i^{sub}. □

Theorem 4. Consider query set Q_i and its graph model $G(V, E)$. Let $MCS(G)$ denote the maximum clique size of G. It holds that:

$$\Delta_{Q_i} \leq MCS(G) \tag{5.2}$$

Proof. We prove by contradiction. Assume that $\Delta_{Q_i} = MCS(G) + \beta$, where β is a positive integer. Then, by Definition 2, there must exist a location $\ell_u \in \mathcal{L}$ such that user u with location ℓ_u answers positively to $MCS(G) + \beta$ queries. If such ℓ_u exists, then by Equation 3.1 there exist $MCS(G) + \beta$ many queries whose A_q contain ℓ_u, meaning that they have a non-empty common intersection encompassing ℓ_u. However, according to Lemma 1, if these queries have a non-empty common intersection, then the vertices representing these queries must form a clique with size $|MCS(G) + \beta|$. Yet, by definition this is impossible because there cannot exist a clique with size greater than the maximum clique size $MCS(G)$. Hence, contradiction. □

Algorithm 3: Find Sensitivity

Input : Query set Q_i

Output: Sensitivity Δ_{Q_i}

1 $G \leftarrow$ GRAPH_MODEL(Q_i)

2 $mcs \leftarrow$ CALC_MAX_CLIQUE_SIZE(G)

3 **return** mcs

Leveraging Theorem 4, we construct our FIND_SENSITIVITY function as shown in Algorithm 3. Given a query set Q_i for which sensitivity must be computed, the algorithm first converts Q_i to its graph model G. Then, the maximum clique size of G is computed and returned as output.

We can exemplify this approach using Figure 5.1. Assume that the leftmost set of 6 queries is given as input to Algorithm 3. For the graph model of this query set, its maximum clique consists of $\{q_4, q_5, q_6\}$ and therefore $MCS(G)$=3. Indeed, Δ_{Q_i} is also 3 since the joint common intersection with maximum cardinality is the joint intersection of $\{q_4, q_5, q_6\}$, as can be seen from the leftmost portion of Figure 5.1.

5.5 User-Side Noisy Response Computation

With knowledge of Q_i and Δ_{Q_i}, each user u is now ready to locally compute their noisy response satisfying ε-LDP and send it to the server. This is indeed the goal of the COMPUTE_RESPONSE function, which we describe in this section. We provide three options to choose from: (i) adapted version of Basic RAPPOR, (ii) OBE from Chapter 4, and (iii) binary version of Randomized Response. Next, we explain how each of these 3 options are used in our setting.

Basic RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response (RAPPOR) is a protocol developed by Google in order to retrieve user statistics from Chrome [Erlingsson et al., 2014]. In the basic version of RAPPOR without the Bloom filter, each user u's true value is assumed to be encoded into a bitvector B_u. Then, a perturbation mechanism Ψ takes B_u as input and outputs a perturbed bitvector B'_u. The perturbed bitvector is sent to the server as the user's

noisy response.

We use Basic RAPPOR for answering spatial density queries as follows. Let $Q_i = \{q_1, q_2, ..., q_m\}$ with sensitivity Δ be the query set that needs to be answered by user u with location ℓ_u. Note that $m = |Q_i|$. Let ε be the privacy parameter. User u initializes bitvector B_u with length m, as:

$$j \in [1, m]: \quad B_u[j] = \begin{cases} 1 & \text{if } \varphi(u, q_j) = 1 \\ 0 & \text{if } \varphi(u, q_j) = 0 \end{cases} \tag{5.3}$$

Afterwards, perturbation mechanism Ψ takes B_u as input and outputs a perturbed bitvector B'_u with probabilities:

$$\Pr\Big[B'_u[j] = \Psi(B_u[j]) = 1\Big] = \begin{cases} \frac{e^{\varepsilon/2\Delta}}{e^{\varepsilon/2\Delta}+1} & \text{if } B_u[j] = 1 \\ \frac{1}{e^{\varepsilon/2\Delta}+1} & \text{if } B_u[j] = 0 \end{cases} \tag{5.4}$$

User u sends perturbed bitvector B'_u to the server.

Optimized Bitvector Encoding (OBE): OBE is an extended version of the OUE protocol [Wang et al., 2017], which we proposed in Chapter 4. Similar to RAPPOR, it also uses bitvector encoding and bit-by-bit perturbation. However, the perturbation mechanism Ψ is designed differently in OBE, to improve estimation accuracy (while still satisfying ε-LDP).

We use OBE for answering spatial density queries as follows. Similar to RAPPOR, user u initializes bitvector B_u with length m the same way as in Equation 5.3. Afterwards, perturbation mechanism Ψ of OBE takes B_u as input and outputs a perturbed bitvector B'_u with probabilities:

$$\Pr\Big[B'_u[j] = \Psi(B_u[j]) = 1\Big] = \begin{cases} \frac{1}{2} & \text{if } B_u[j] = 1 \\ \frac{1}{e^{\varepsilon/\Delta}+1} & \text{if } B_u[j] = 0 \end{cases} \tag{5.5}$$

User u sends perturbed bitvector B'_u to the server.

Binary Randomized Response (BRR) is the binary version of the Generalized Randomized Response protocol [Wang et al., 2017, Gursoy et al., 2019]. It is suitable when the user's true response can be encoded into one bit, i.e., $v \in \{0, 1\}$. Because of this property, in the context of answering spatial density queries, BRR

can be used when $|Q_i| = 1$, e.g., in singleton partitioning. Indeed, it provides high server-side estimation accuracy when $|Q_i| = 1$.

Let $Q_i = \{q_1\}$ be the query set that needs to be answered by user u with location ℓ_u, and let ε be the privacy parameter. User's true response v is a single bit that is determined as:

$$v = \begin{cases} 1 & \text{if } \varphi(u, q_1) = 1 \\ 0 & \text{if } \varphi(u, q_1) = 0 \end{cases} \tag{5.6}$$

The perturbation mechanism Ψ of BRR takes v as input and produces a 0 or 1 bit as output, with probability:

$$\Pr[\Psi(v) = 1] = \begin{cases} \frac{e^\varepsilon}{e^\varepsilon + 1} & \text{if } v = 1 \\ \frac{1}{e^\varepsilon + 1} & \text{if } v = 0 \end{cases} \tag{5.7}$$

The output bit is sent to the server.

Example: To demonstrate user-side noisy response computation, we give a visual example in Figure 5.2. Since BRR is simpler, our example is with bitvector encoding (applicable to RAPPOR or OBE). Consider the query set $Q_i = \{q_1, q_2, q_3, q_4, q_5, q_6\}$ given in Figure 5.2 and two users' locations: Bob and Charlie. B_{Bob} has the first two bits set to 1 while the rest are 0, since $\varphi(Bob, q_1) = 1$ and $\varphi(Bob, q_2) = 1$, but for the rest of the positions, $\varphi(Bob, q_j) = 0$. Then, Bob's bitvector is perturbed bit-by-bit on Bob's device to obtain B'_{Bob}, which is sent to the server (data collector).

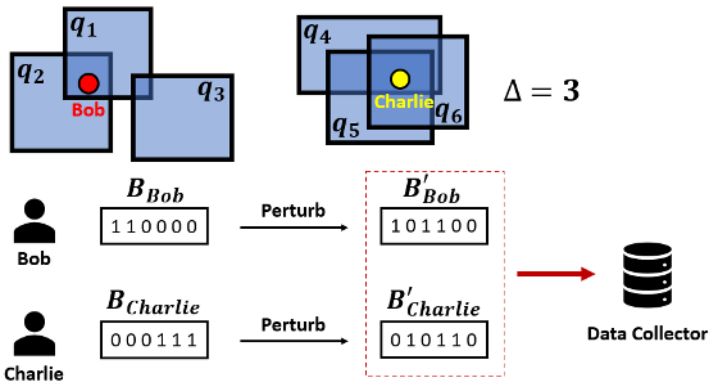

Figure 5.2: Sample user-side noisy response computation for two users: Bob and Charlie

5.6 Server-Side Estimation

After each user computes and sends their response to the server using LDP, the server collects these responses and performs estimation in two steps. The first step depends on which protocol is being used (RAPPOR, OBE or BRR). The second step is applied the same way regardless of the protocol.

Consider query set $Q_i = \{q_1, q_2, ..., q_m\}$ where $m = |Q_i|$. LDP responses from population P_i are available at the time of estimation. The goal is to estimate $ans_{Q_i} = \{ans_{q_1}, ans_{q_2}, ..., ans_{q_m}\}$, that is, ans_{q_j} for all $j \in [1, m]$. To achieve this goal, in the first step, we derive an initial estimate $\bar{C}(j)$ which depends on the protocol. In the second step, we derive ans_{q_j} from $\bar{C}(j)$ in a protocol-agnostic manner.

Estimation of $\bar{C}(j)$ for RAPPOR: When estimating $\bar{C}(j)$, the server makes use of how many users in P_i sent a positive bit (1 bit) in the corresponding position of their perturbed bitvector $B'_u[j]$. Then, the initial estimate $\bar{C}(j)$ is computed as:

$$\bar{C}(j) = \frac{\Big(\sum\limits_{u \in P_i} B'_u[j]\Big) - \alpha \cdot |P_i|}{1 - 2\alpha} \tag{5.8}$$

where α is the bit flipping probability: $\alpha = \frac{1}{e^{\varepsilon/2\Delta}+1}$. Following [Erlingsson et al., 2014, Wang et al., 2017], it is straightforward to show $\bar{C}(j)$ is an unbiased estimate.

Estimation of $\bar{C}(j)$ for OBE: Similar to RAPPOR, the responses that are collected from users are bitvectors. Again, using how many users in P_i sent a positive bit in the corresponding position of their $B'_u[j]$, the server initially estimates $\bar{C}(j)$ as:

$$\bar{C}(j) = \frac{2 \cdot \Big[(e^{\varepsilon/\Delta} + 1) \cdot \Big(\sum\limits_{u \in P_i} B'_u[j]\Big) - |P_i|\Big]}{e^{\varepsilon/\Delta} - 1} \tag{5.9}$$

Following Theorem 2, $\bar{C}(j)$ provides an unbiased estimate.

Estimation of $\bar{C}(j)$ for BRR: In BRR, the LDP responses that the server collects from users are single bits as opposed to bitvectors. Let b_u denote the bit that was collected from user u. The server computes $\bar{C}(j)$ as:

$$\bar{C}(j) = \frac{\Big(\sum\limits_{u \in P_i} b_u\Big) \cdot (e^{\varepsilon} + 1) - |P_i|}{e^{\varepsilon} - 1} \tag{5.10}$$

Derivation of ans_{q_j} from $\bar{C}(j)$: This step applies the same way to all protocols (RAPPOR, OBE, BRR). It converts the initial estimates $\bar{C}(j)$ to ans_{q_j} as follows:

$$ans_{q_j} = \begin{cases} \bar{C}(j) \cdot \frac{|\mathcal{P}|}{|P_i|} & \text{if } \bar{C}(j) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.11}$$

There are two main reasons why this conversion is used. First reason is to enforce non-negativity. Due to the randomized perturbation of LDP, some estimates may turn out to be negative (below 0). However, in truth, spatial densities are always non-negative. Thus, we post-process negative densities to 0. Second reason is due to partitioning. By nature of partitioning, the server asks q_j to a subset of the population $P_i \subseteq \mathcal{P}$. Thus, although the initial estimate $\bar{C}(j)$ is computed with respect to P_i, we need to amplify the result to $\mathcal{P}$ so that the server estimates ans_{q_j} for the whole population $\mathcal{P}$. We achieve this by multiplying $\bar{C}(j)$ with $\frac{|\mathcal{P}|}{|P_i|}$.

Chapter 6

EXPERIMENTAL EVALUATION

We implemented our solution in Python, including all partitioning strategies and protocols. For graph modeling and graph-related computations (e.g., finding max clique), the *networkx* library is used. In this chapter, we evaluate our solution experimentally and compare the utility of the partitioning strategies and protocols.

6.1 Experiment Setup

Datasets: In our experiments, we used 4 independent, publicly available, real-world location datasets: Kaggle, Foursquare, Brightkite, Gowalla. Each dataset was processed so that each user has one location ℓ_u consisting of latitude (lat) and longitude (lon) coordinates.

Kaggle is a taxi trajectory dataset which was originally made public for the taxi service prediction competition in ECML-PKDD 2015. We obtained the dataset from Kaggle [ECML/PKDD, 2015]. It contains taxis' trajectories in the span of a year in Porto, separated into trips. We processed the data so that each trajectory is reduced into the starting point of the trip, i.e., each user u's location ℓ_u corresponds to the location from which they started their taxi trip. At the end of the processing, our dataset contained 1,303,485 users with their corresponding locations. The minimum latitude was 41.13, maximum latitude was 41.19, minimum longitude was -8.63, and maximum longitude was -8.57.

Foursquare contains location check-ins of users in the span of 10 months from the city of Tokyo [Yang et al., 2015]. We used this dataset as is. It contains data of 573,703 users. The minimum latitude was 35.51, maximum latitude was 35.87, minimum longitude was 139.47, and maximum longitude was 139.91.

Brightkite dataset contains users' location check-ins from an old location-based

social network service provider called Brightkite [Cho et al., 2011]. Although the dataset contains check-ins from users worldwide, we found that the majority of the world contains sparse check-ins, whereas the north-eastern portion of the USA contained higher number of check-ins. Thus, we reduced the original dataset into the north-eastern portion of the USA. At the end of this reduction, our dataset contained 151,139 users and their locations. The minimum latitude was 40.58, maximum latitude was 40.85, minimum longitude was -74.51, and maximum longitude was -72.93.

Gowalla was also a location-based social networking site where users shared their locations via check-ins [Cho et al., 2011]. Similar to Brightkite, we reduced the original Gowalla dataset using the same min/max latitude and longitude coordinates in order to focus on the denser north-eastern portion of the USA. At the end, our dataset contained 146,012 users and their locations.

Compared Approaches: Our solution can work with multiple protocols (OBE, RAPPOR, BRR) and partitioning strategies (Singleton, Holistic, Random, Advanced) apart from special cases, e.g., BRR requires $|Q_i| = 1$ and therefore it is only usable with Singleton partitioning. We abbreviate the four partitioning strategies as: SING for Singleton, HOL for Holistic, RAND for Random, ADV for Advanced. Considering their possible combinations with the protocols, we have a total of 8 approaches that we compare experimentally: OBE-SING, OBE-HOL, OBE-RAND, OBE-ADV, RAPPOR-HOL, RAPPOR-RAND, RAPPOR-ADV, BRR-SING. We note that for RAND, since the number of partitions n is variable, we repeat the experiments by varying n and report the best results in our graphs and tables. This serves to show that when our proposed ADV strategy is better than RAND, it is indeed better for all possible n. Additionally, since all of the protocols perturb reported bitvectors probabilistically, we repeat each experiment 10 times and compare the average errors of these experiments. This provides a better representation of the actual performance of each setting. We also note that the HOL approach corresponds to straightforward application of [Inan et al., 2020] to LDP, and the fact that ADV outperforms HOL shows the benefit in partitioning.

Query Sets: In order to perform experiments with varying query set sizes and characteristics, we implemented a query generation algorithm. Our algorithm constructs query set $\mathcal{Q}$ by generating rectangular-shaped queries in randomized fashion. For each query q, first the midpoint of A_q is decided by selecting its latitude and longitude uniformly at random from $\mathcal{L}$. Then, we use a parameter μ to control the expected size of A_q, i.e., if $\mu = 4$, the size of A_q is $4\% \times |\mathcal{L}|$. To add randomness to query sizes, instead of directly constructing A_q with size $\mu \times |\mathcal{L}|$ around the midpoint; we draw a random sample between $(\mu - 1\%) \times |\mathcal{L}|$ and $(\mu + 1\%) \times |\mathcal{L}|$, and make the size of A_q equal to this sample. Following this approach, we generate $\mathcal{Q}$ for various $|\mathcal{Q}|$ such as $|\mathcal{Q}| = 200, 400, ..., 1000$.

Error Metrics: For query set $\mathcal{Q}$, let $ans_{\mathcal{Q}} = \{ans_{q_1}, ans_{q_2}, ans_{q_3}, ..., ans_{|\mathcal{Q}|}\}$ denote the ground truth answers to queries in $\mathcal{Q}$ and let $\overline{ans_{\mathcal{Q}}} = \{\overline{ans_{q_1}}, \overline{ans_{q_2}}, \overline{ans_{q_3}}, ..., \overline{ans_{|\mathcal{Q}|}}\}$ denote the answers estimated at the end of an LDP approach. We measure the error in $\overline{ans_{\mathcal{Q}}}$ using three error metrics: Absolute error, RMSE and Kendall-tau. While absolute error and RMSE measures error by directly comparing ans_{q_i} and $\overline{ans_{q_i}}$, Kendall-tau measures how well the relative rankings of query answers are preserved, e.g., if $ans_{q_i} > ans_{q_j}$, does it hold that $\overline{ans_{q_i}} > \overline{ans_{q_j}}$? Such relative rankings would be important to preserve for applications such as restaurant popularity (e.g., preserving the fact that restaurant X is more crowded than restaurant Y) or traffic navigation (e.g., preserving the fact that road X is more crowded than road Y).

Mean Absolute Error (MAE) calculates the average absolute difference between the estimated and the ground truth answers:

$$MAE = \frac{\sum_{i=1}^{|\mathcal{Q}|} |ans_{q_i} - \overline{ans_{q_i}}|}{|\mathcal{Q}|} \tag{6.1}$$

Root Mean Squared Error (RMSE) is a popular error measurement metric, formally defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{|\mathcal{Q}|} (ans_{q_i} - \overline{ans_{q_i}})^2}{|\mathcal{Q}|}} \tag{6.2}$$

Kendall-tau (KT) measures how well the relative rankings of top-k query answers are preserved. Consider k queries from $\mathcal{Q}$ whose ground truth answers are highest. Among them, a pair of queries q_i, q_j are said to be concordant if either of the following holds. Otherwise, q_i, q_j are said to be discordant.

$$ans_{q_i} > ans_{q_j} \quad \wedge \quad \overline{ans_{q_i}} > \overline{ans_{q_j}}$$
$$ans_{q_i} < ans_{q_j} \quad \wedge \quad \overline{ans_{q_i}} < \overline{ans_{q_j}}$$

The total number of concordant pairs and discordant pairs can be calculated across all pairs of queries q_i, q_j among the top-k. Then, the KT metric is defined as:

$$KT = \frac{(\#\text{ of concordant pairs}) - (\#\text{ of discordant pairs})}{k(k-1)/2} \tag{6.3}$$

By default, we use $k = 200$ in our experiments.

6.2 Comparison Between Different Protocols and Partitioning Strategies

In Tables 6.1, 6.2 and 6.3, we compare all 8 approaches side-by-side. The comparisons are done with all 4 datasets and with ε varying between 0.1 and 2.0. Table 6.1 uses the RMSE metric, Table 6.2 uses the MAE metric and Table 6.3 uses the KT metric. As expected, when ε increases, generally RMSE and MAE decreases (lower RMSE and MAE are better) and KT increases (higher KT is better).

There are two main highlights according to these results. First is the superiority of our proposed Advanced Partitioning strategy over other partitioning strategies. According to Table 6.1, out of 16 total comparison settings, OBE-ADV performs best in 9 of them and RAPPOR-ADV performs best in 4 of them, both of which use Advanced Partitioning. They are only beaten by BRR-SING when ε is relatively high (such as $\varepsilon = 1$ or 2) and on the two smaller datasets (Brightkite and Gowalla). Furthermore, Table 6.2 shows that out of 16 total comparison settings, OBE-ADV performs best in 11 of them, RAPPOR-ADV performs best in 3 of them, and for 1 setting (Gowalla database, $\varepsilon = 1.0$) OBE-ADV and RAPPOR-ADV both perform best. Similarly, according to Table 6.3, out of 16 total comparison settings, OBE-ADV performs best in 11 of them and RAPPOR-ADV performs best in 3 of them.

As a whole, these results show that our proposed Advanced Partitioning strategy is indeed the best-performing option among all partitioning strategies. The second highlight is the comparison between OBE-ADV and RAPPOR-ADV. While both of them use Advanced Partitioning, in most cases OBE-ADV is more likely to yield lower error compared to RAPPOR-ADV, demonstrating the benefit of OBE.

It is also worth noting that the Holistic Partitioning approaches (RAPPOR-HOL and OBE-HOL) are typically the worst performers, having higher error than Singleton Partitioning and Random Partitioning. This supports our earlier analysis from Section 5.2 that noise error is more dominant than sampling error, since Holistic Partitioning is the strategy that incurs heaviest noise error (it has largest Δ among all). The results clearly demonstrate that using *some* type of partitioning is better than the Holistic approach, which justifies our overall design decision to partition the query and user population into subsets rather than requiring all users to answer all queries.

Table 6.1: RMSE results under 4 different ε values and 4 datasets. $|\mathcal{Q}| = 600$ and $\mu = 4$. Lowest RMSE in each row (best result) is marked in bold. Results show that Advanced Partitioning (RAPPOR-ADV and OBE-ADV) yields best results in 13 out of 16 cases. 9 of them are with the proposed protocol OBE.

		BRR-SING	**RAPPOR-HOL**	**RAPPOR-RAND**	**RAPPOR-ADV**	**OBE-SING**	**OBE-HOL**	**OBE-RAND**	**OBE-ADV**
Kaggle	$\varepsilon = 0.1$	2.03×10^5	6.53×10^5	2.81×10^5	1.10×10^5	3.98×10^5	6.54×10^5	3.27×10^5	1.08×10^5
	$\varepsilon = 0.5$	4.63×10^4	1.35×10^5	6.32×10^4	2.71×10^4	8.58×10^4	1.38×10^5	7.02×10^4	2.57×10^4
	$\varepsilon = 1.0$	2.38×10^4	7.07×10^4	3.38×10^4	1.57×10^4	4.43×10^4	7.20×10^4	3.75×10^4	1.54×10^4
	$\varepsilon = 2.0$	1.22×10^4	3.82×10^4	1.71×10^4	1.16×10^4	2.28×10^4	3.71×10^4	1.88×10^4	1.09×10^4
Foursquare	$\varepsilon = 0.1$	1.33×10^5	4.48×10^5	1.86×10^5	7.01×10^4	2.62×10^5	4.45×10^5	2.32×10^5	7.30×10^4
	$\varepsilon = 0.5$	2.73×10^4	9.36×10^4	3.98×10^4	1.63×10^4	5.43×10^4	9.29×10^4	4.61×10^4	1.61×10^4
	$\varepsilon = 1.0$	1.47×10^4	4.67×10^4	2.08×10^4	8.78×10^3	2.80×10^4	4.74×10^4	2.41×10^4	8.57×10^3
	$\varepsilon = 2.0$	7.80×10^3	2.44×10^4	1.07×10^4	5.40×10^3	1.34×10^4	2.45×10^4	1.20×10^4	4.78×10^3
Brightkite	$\varepsilon = 0.1$	6.55×10^4	2.13×10^5	9.51×10^4	3.52×10^4	1.33×10^5	2.15×10^5	1.10×10^5	3.60×10^4
	$\varepsilon = 0.5$	1.41×10^4	4.37×10^4	1.98×10^4	1.05×10^4	1.34×10^4	2.21×10^4	1.14×10^4	9.36×10^3
	$\varepsilon = 1.0$	6.95×10^3	2.22×10^4	9.86×10^3	8.29×10^3	1.34×10^4	2.21×10^4	1.14×10^4	7.72×10^3
	$\varepsilon = 2.0$	3.40×10^3	1.14×10^4	4.97×10^3	7.51×10^3	6.52×10^3	1.14×10^4	5.67×10^3	7.82×10^3
Gowalla	$\varepsilon = 0.1$	6.58×10^5	2.10×10^5	9.28×10^4	3.47×10^4	1.31×10^5	2.09×10^5	1.15×10^5	3.40×10^4
	$\varepsilon = 0.5$	1.37×10^4	4.21×10^4	1.94×10^4	7.58×10^3	2.78×10^4	4.18×10^4	2.24×10^4	7.70×10^3
	$\varepsilon = 1.0$	6.74×10^3	2.09×10^4	9.52×10^3	4.87×10^3	1.32×10^4	2.08×10^4	1.11×10^4	4.97×10^3
	$\varepsilon = 2.0$	3.31×10^3	1.03×10^4	4.88×10^3	4.02×10^3	6.42×10^3	1.07×10^4	5.53×10^3	3.64×10^3

Table 6.2: MAE results under 4 different ε values and 4 datasets. $|\mathcal{Q}| = 600$ and $\mu = 4$. Lowest MAE in each row (best result) is marked in bold. Results show that Advanced Partitioning (RAPPOR-ADV and OBE-ADV) yields best results in 15 out of 16 cases. 12 of them are with the proposed protocol OBE.

		BRR-SING	RAPPOR-HOL	RAPPOR-RAND	RAPPOR-ADV	OBE-SING	OBE-HOL	OBE-RAND	OBE-ADV
Kaggle	$\varepsilon = 0.1$	1.32×10^5	3.93×10^5	1.78×10^5	7.60×10^4	2.47×10^5	3.95×10^5	1.98×10^5	7.42×10^4
	$\varepsilon = 0.5$	3.42×10^4	9.17×10^5	4.56×10^4	2.03×10^4	6.11×10^4	9.31×10^5	4.89×10^4	1.92×10^4
	$\varepsilon = 1.0$	1.81×10^4	5.11×10^4	2.49×10^4	1.16×10^4	3.24×10^4	5.12×10^4	2.68×10^4	1.13×10^4
	$\varepsilon = 2.0$	9.29×10^3	2.83×10^4	1.31×10^4	7.43×10^3	1.71×10^4	2.75×10^4	1.36×10^4	6.97×10^3
Foursquare	$\varepsilon = 0.1$	8.21×10^4	2.63×10^5	1.14×10^5	4.57×10^4	1.58×10^5	2.61×10^5	1.30×10^5	4.71×10^4
	$\varepsilon = 0.5$	1.93×10^4	5.97×10^4	2.72×10^4	1.19×10^4	3.60×10^4	5.96×10^4	2.99×10^4	1.18×10^4
	$\varepsilon = 1.0$	1.07×10^4	3.11×10^4	1.50×10^4	6.65×10^3	1.97×10^4	3.17×10^4	1.64×10^4	6.46×10^3
	$\varepsilon = 2.0$	5.85×10^3	1.74×10^4	8.06×10^3	3.93×10^3	1.00×10^4	1.75×10^4	8.55×10^3	3.56×10^3
Brightkite	$\varepsilon = 0.1$	3.81×10^4	1.23×10^5	5.55×10^4	2.14×10^4	7.64×10^4	1.24×10^5	6.25×10^4	2.20×10^4
	$\varepsilon = 0.5$	8.87×10^3	2.61×10^4	1.24×10^4	6.02×10^3	1.67×10^4	2.65×10^4	1.33×10^4	5.62×10^3
	$\varepsilon = 1.0$	4.56×10^3	1.35×10^4	6.40×10^3	4.11×10^3	8.41×10^3	1.36×10^4	6.89×10^3	3.96×10^3
	$\varepsilon = 2.0$	2.36×10^3	7.32×10^3	3.37×10^3	3.20×10^3	4.27×10^3	7.15×10^3	3.59×10^3	3.17×10^3
Gowalla	$\varepsilon = 0.1$	3.85×10^4	1.21×10^5	5.38×10^4	2.08×10^4	7.63×10^4	1.20×10^5	6.31×10^4	2.05×10^4
	$\varepsilon = 0.5$	8.53×10^3	2.52×10^4	1.18×10^4	4.76×10^3	1.61×10^4	2.50×10^4	1.30×10^4	4.90×10^3
	$\varepsilon = 1.0$	4.34×10^3	1.27×10^4	6.02×10^3	2.88×10^3	8.17×10^3	1.26×10^4	6.68×10^3	2.88×10^3
	$\varepsilon = 2.0$	2.22×10^3	6.44×10^3	3.25×10^3	2.01×10^3	4.12×10^3	6.74×10^3	3.43×10^3	1.86×10^3

Table 6.3: KT results under 4 different ε values and 4 datasets. $|\mathcal{Q}| = 600$, $\mu = 4$ and $k = 200$. Highest KT in each row (best result) is marked in bold. Results show that OBE-ADV yields best results in 11 out of 16 cases.

		BRR-SING	RAPPOR-HOL	RAPPOR-RAND	RAPPOR-ADV	OBE-SING	OBE-HOL	OBE-RAND	OBE-ADV
Kaggle	$\varepsilon = 0.1$	0.449	0.200	0.333	0.575	0.265	0.126	0.335	0.655
	$\varepsilon = 0.5$	0.911	0.581	0.836	0.950	0.739	0.585	0.812	0.952
	$\varepsilon = 1.0$	0.957	0.833	0.943	0.962	0.907	0.801	0.925	0.967
	$\varepsilon = 2.0$	0.970	0.936	0.972	0.969	0.958	0.943	0.973	0.970
Foursquare	$\varepsilon = 0.1$	0.232	0.136	0.227	0.463	0.217	0.119	0.177	0.515
	$\varepsilon = 0.5$	0.713	0.413	0.622	0.871	0.566	0.404	0.542	0.886
	$\varepsilon = 1.0$	0.895	0.557	0.839	0.936	0.741	0.578	0.741	0.944
	$\varepsilon = 2.0$	0.939	0.764	0.937	0.960	0.898	0.787	0.913	0.955
Brightkite	$\varepsilon = 0.1$	0.214	0.027	0.142	0.261	0.114	0.046	0.159	0.280
	$\varepsilon = 0.5$	0.423	0.254	0.391	0.495	0.348	0.252	0.375	0.423
	$\varepsilon = 1.0$	0.553	0.320	0.545	0.574	0.361	0.332	0.467	0.577
	$\varepsilon = 2.0$	0.754	0.507	0.643	0.665	0.575	0.485	0.663	0.667
Gowalla	$\varepsilon = 0.1$	0.162	0.066	0.156	0.278	0.084	0.053	0.138	0.253
	$\varepsilon = 0.5$	0.386	0.268	0.357	0.521	0.315	0.186	0.325	0.539
	$\varepsilon = 1.0$	0.555	0.314	0.447	0.645	0.369	0.350	0.402	0.595
	$\varepsilon = 2.0$	0.722	0.399	0.614	0.743	0.540	0.384	0.600	0.766

6.3 Impact of $|\mathcal{Q}|$

In the next set of experiments, we analyze the impact of query set size $|\mathcal{Q}|$ on RMSE and KT. For this purpose, we fix $\varepsilon = 1$ and $\mu = 4$, and vary the query set size $|\mathcal{Q}|$.

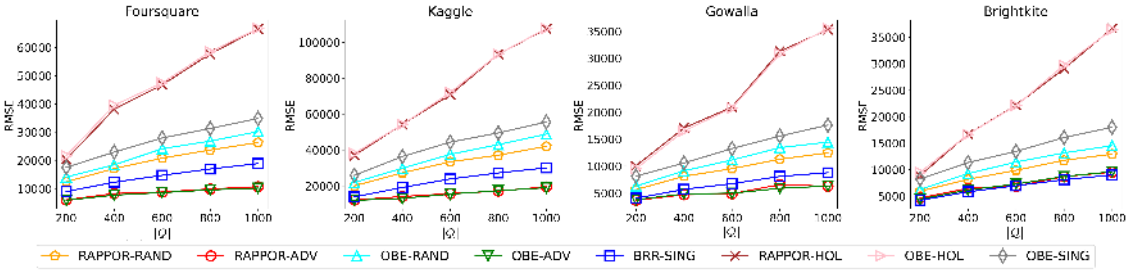

Figure 6.1: RMSE results with $\varepsilon = 1$, $\mu = 4$, and varying $|\mathcal{Q}|$.

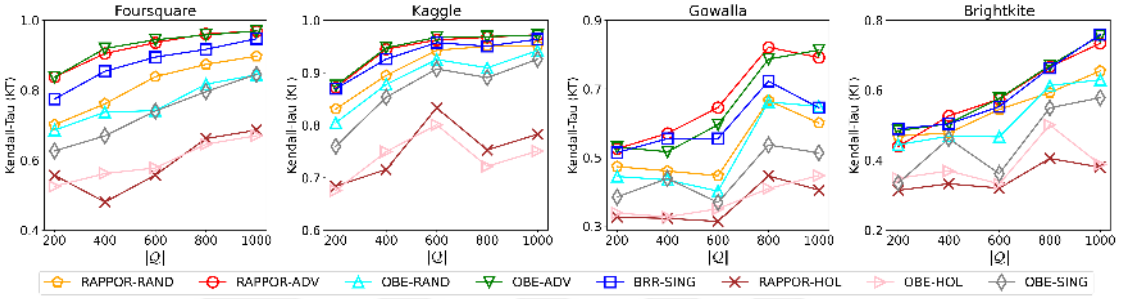

Figure 6.2: KT results with $\varepsilon = 1$, $\mu = 4$, $k = 200$, and varying $|\mathcal{Q}|$.

RMSE results are reported in Figure 6.1 and KT results are reported in Figure 6.2. Overall, results show that OBE-ADV and RAPPOR-ADV remain as the best choices (lowest RMSE and highest KT), followed oftentimes by BRR-SING, then the two Random Partitioning approaches (RAPPOR-RAND and OBE-RAND). The Holistic Partitioning approaches are once again the least desired.

In Figure 6.1, we observe that as $|\mathcal{Q}|$ increases, RMSE values of all approaches tend to increase. The reason behind this observation is different from approach to approach. In case of RAPPOR-HOL and OBE-HOL, increased $|\mathcal{Q}|$ means that Δ will be higher; thus, the perturbation probabilities and consequently the RMSE values also increase. In case of the Singleton Partitioning approaches (BRR-SING and OBE-SING), since the number of partitions is equal to $|\mathcal{Q}|$, increased $|\mathcal{Q}|$ implies higher number of partitions and therefore there are fewer users per partition to answer the queries. As a result, the sampling error (as discussed in Section 5.2) increases. In case of the Advanced Partitioning approaches (OBE-ADV and RAPPOR-ADV), increased $|\mathcal{Q}|$ causes larger number of overlaps and intersecting

vertices; thus, Algorithm 2 typically produces a higher number of partitions. Similar to earlier, this causes sampling error to increase.

In Figure 6.2, we observe that as $|\mathcal{Q}|$ increases, KT values also tend to increase, but not as smoothly as RMSE. There are two conflicting factors at play here. On one hand, as shown by RMSE results, queries' errors generally increase when $|\mathcal{Q}|$ increases, therefore we would expect KT to become lower. On the other hand, since the KT metric measures rankings of top $k = 200$, as $|\mathcal{Q}|$ increases, the queries that are being measured by the KT metric become the topmost popular queries among a larger $|\mathcal{Q}|$. For example, when $|\mathcal{Q}| = 200$, the KT metric is measured over 100% of the queries, but when $|\mathcal{Q}| = 1000$, the KT metric is measured over the 20% most popular queries with highest ans_q. Since popular queries have high answers by definition, despite the increased errors, they remain more likely to preserve their relative rankings. Combining these two factors, we arrive at the non-smooth trends observed in Figure 6.2. It should be noted that despite the non-smooth trends, the proposed OBE-ADV approach still seems to be the best among all approaches, followed by RAPPOR-ADV. As one exception, on the Brightkite dataset we obtain BRR-SING performing very close to, or slightly better than, RAPPOR-ADV and OBE-ADV in Figures 6.1 and 6.2.

6.4 Impact of μ

Finally, we conduct a set of experiments to analyze the impact of queries' geographical region sizes (μ) on RMSE and KT. For this purpose, we fix $\varepsilon = 1$ and $|\mathcal{Q}| = 600$, and change μ. The different μ values we use include 1, 2, 4, 6, 8 and 10; ranging from small queries in which A_q is $1\% \times |\mathcal{L}|$, to relatively large queries in which A_q is $10\% \times |\mathcal{L}|$. RMSE results are reported in Figure 6.3 and KT results are reported in Figure 6.4.

Overall, results show that OBE-ADV and RAPPOR-ADV remain as the top-2 best performing approaches. The reasons for the positive correlations between μ and RMSE, as observed in Figure 6.3, are similar to the previous section. For Holistic Partitioning and Random Partitioning approaches, increased μ has a direct impact

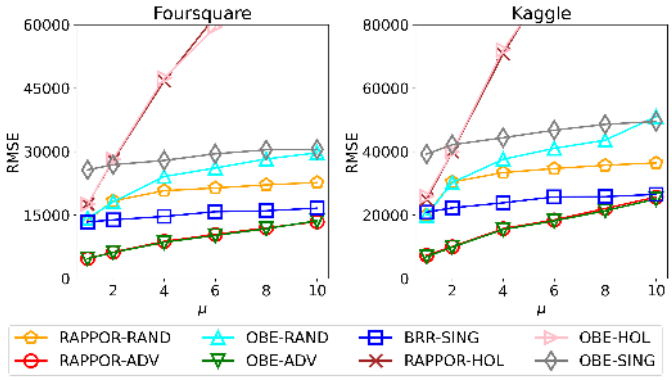

Figure 6.3: RMSE results with $\varepsilon = 1$, $|\mathcal{Q}| = 600$, and varying μ.

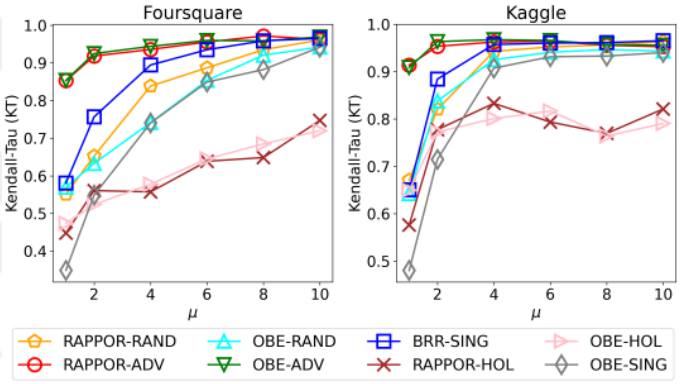

Figure 6.4: KT results with $\varepsilon = 1$, $|\mathcal{Q}| = 600$, and varying μ.

on increased Δ, since larger A_q have higher likelihood of intersecting. Thus, with increased μ, we see a clear increase in the RMSE of Holistic Partitioning and Random Partitioning approaches. In case of Advanced Partitioning, more intersections causes Algorithm 2 to produce a larger number of partitions. Therefore, due to sampling error, RAPPOR-ADV and OBE-ADV's RMSE values increase with μ.

Figure 6.4 shows that across all of the approaches, there are positive correlations between KT and μ. This is because with increased μ, larger A_q causes ans_q to typically become larger, and therefore the magnitude of the differences between ans_{q_i} and ans_{q_j} become larger. As a result, despite the noise caused by LDP, it is easier to preserve the fact that ans_{q_i} is relatively larger (or smaller) than ans_{q_j}. This causes a positive impact on KT. However, if one focuses on the differences between the approaches (e.g., RAPPOR-HOL vs BRR-SING vs OBE-ADV), these

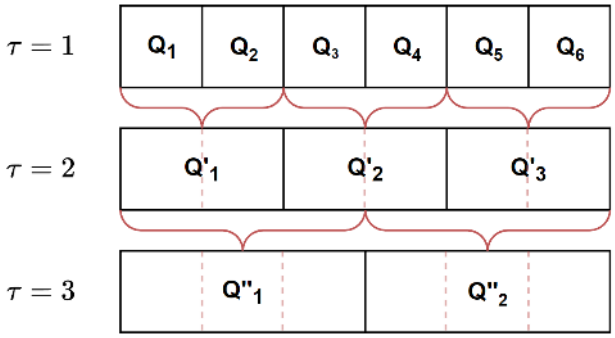

Figure 6.5: Corresponding query subsets for $\tau = 1$, $\tau = 2$ and $\tau = 3$

differences are more visible when μ is small. This is because with large μ, KT values of all approaches start converging to 1 (perfect KT score), due to the relative ease of preserving rankings for higher μ. Thus, when μ is small like 2 or 4, the superiority of Advanced Partitioning is clear; but as μ approaches 10, many of the approaches converge to each other in terms of KT results.

6.5 Comparison of Noise Error and Sampling Error

In this section, we extend the existing Advanced Partitioning method by introducing an additional parameter called sensitivity threshold τ. The rationale behind our extension is to demonstrate the relative impact of noise error versus sampling error, thereby exemplifying the discussion in Section 5.2 empirically.

Recall from Section 5.3 that Advanced Partitioning guarantees $\Delta_{Q_i} = 1$ for each query subset Q_i. This way, Advanced Partitioning minimizes the noise error by minimizing the sensitivity of query subsets. However, it is also possible to give Advanced Partitioning more flexibility by limiting sensitivity to a higher value $\Delta_{Q_i} \leq \tau$. We achieve this by iteratively merging query subsets which are outputs of the original Advanced Partitioning method. Figure 6.5 illustrates the idea behind our merging process: When $\tau = 1$, the output of standard Advanced Partitioning is 6 query partitions $Q_1, Q_2, ..., Q_6$. When $\tau = 2$, these query partitions are merged in pairs such that $Q'_1 = Q_1 \cup Q_2$, $Q'_2 = Q_3 \cup Q_4$ and $Q'_3 = Q_5 \cup Q_6$. When $\tau = 3$, $Q''_1 = Q_1 \cup Q_2 \cup Q_3$ and $Q''_2 = Q_4 \cup Q_5 \cup Q_6$.

Table 6.4: Evaluating the extension to Advanced Partitioning. OBE protocol, MAE results with $\varepsilon = 1$, $|\mathcal{Q}| = 600$, $\mu = 4$, and varying τ.

Kaggle	$\tau = 1$	1.13×10^4
	$\tau = 2$	1.51×10^4
	$\tau = 3$	1.73×10^4
Foursquare	$\tau = 1$	6.46×10^3
	$\tau = 2$	8.70×10^3
	$\tau = 3$	1.05×10^4
Brightkite	$\tau = 1$	3.96×10^3
	$\tau = 2$	4.47×10^3
	$\tau = 3$	4.91×10^3
Gowalla	$\tau = 1$	2.88×10^3
	$\tau = 2$	3.61×10^3
	$\tau = 3$	4.17×10^3

Overall, this extension increases noise error since sensitivity increases. However, the total number of query subsets n decreases, resulting in each query being answered by more users ($|\mathcal{P}|/n$). Therefore, sampling error is reduced.

After implementing the extension described above, we performed a set of experiments to analyze the impact of τ on MAE. For this purpose, we fix $\varepsilon = 1$, $\mu = 4$, $Q = 600$ and change τ. In Table 6.4, we compare different τ values with all 4 datasets for the OBE protocol. The results show that for each of the datasets, as τ increases, MAE also increases. Since increasing τ results in higher noise error but lower sampling error, we observe that noise error is indeed more dominant than sampling error. Therefore, it is shown that the decision of setting $\Delta = 1$ in the original Advanced Partitioning is justified.

Chapter 7

CONCLUSION

In this thesis, we developed methods for answering spatial density queries while preserving LDP. Our solution consists of four main steps: partitioning, finding sensitivity, user-side noisy response computation, and server-side estimation. We proposed three basic partitioning strategies (Singleton, Holistic, Random), and based on their analysis, we developed the Advanced Partitioning strategy which uses vertex coloring on the graph model of a query set to construct more desirable partitions. We then leveraged the graph model of each partition Q_i to find its sensitivity Δ_{Q_i} using maximum clique size. Afterwards, we ensured that each user computes their noisy response using Δ_{Q_i} to satisfy LDP. We developed the user-side noisy response computation and corresponding server-side estimation procedures using three LDP protocols: Basic RAPPOR, BRR and OBE, where OBE is our proposed extension for OUE. Finally, we experimentally demonstrated the benefits of our proposed techniques by showing that Advanced Partitioning (RAPPOR-ADV and OBE-ADV) performs best in the majority of the experiments, with OBE-ADV performing slightly better than RAPPOR-ADV.

BIBLIOGRAPHY

[app, 2020] (2020). Apple – learning with privacy at scale. `https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf`.

[Acs and Castelluccia, 2014] Acs, G. and Castelluccia, C. (2014). A case study: Privacy preserving release of spatio-temporal density in paris. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1679–1688. ACM.

[Arachchige et al., 2019] Arachchige, P. C. M., Bertok, P., Khalil, I., Liu, D., Camtepe, S., and Atiquzzaman, M. (2019). Local differential privacy for deep learning. *IEEE Internet of Things Journal*, 7(7):5827–5842.

[Bassily and Smith, 2015] Bassily, R. and Smith, A. (2015). Local, private, efficient protocols for succinct histograms. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*, pages 127–135. ACM.

[Bassily et al., 2017] Bassily, R., Stemmer, U., Thakurta, A. G., et al. (2017). Practical locally private heavy hitters. In *Advances in Neural Information Processing Systems*, pages 2288–2296.

[Chen et al., 2012a] Chen, R., Acs, G., and Castelluccia, C. (2012a). Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pages 638–649. ACM.

[Chen et al., 2012b] Chen, R., Fung, B., Desai, B. C., and Sossou, N. M. (2012b). Differentially private transit data publication: a case study on the montreal trans-

portation system. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 213–221. ACM.

[Chen et al., 2016] Chen, R., Li, H., Qin, A., Kasiviswanathan, S. P., and Jin, H. (2016). Private spatial data aggregation in the local setting. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 289–300. IEEE.

[Cho et al., 2011] Cho, E., Myers, S. A., and Leskovec, J. (2011). Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 1082–1090, New York, NY, USA. Association for Computing Machinery.

[Cormode et al., 2018a] Cormode, G., Jha, S., Kulkarni, T., Li, N., Srivastava, D., and Wang, T. (2018a). Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1655–1658. ACM.

[Cormode et al., 2018b] Cormode, G., Kulkarni, T., and Srivastava, D. (2018b). Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*, pages 131–146. ACM.

[Cormode et al., 2019] Cormode, G., Kulkarni, T., and Srivastava, D. (2019). Answering range queries under local differential privacy. *Proceedings of the VLDB Endowment*, 12(10):1126–1138.

[Cormode et al., 2012] Cormode, G., Procopiuc, C., Srivastava, D., Shen, E., and Yu, T. (2012). Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering (ICDE)*, pages 20–31. IEEE.

[De Montjoye et al., 2013] De Montjoye, Y.-A., Hidalgo, C. A., Verleysen, M., and Blondel, V. D. (2013). Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports*, 3:1376.

[Ding et al., 2017] Ding, B., Kulkarni, J., and Yekhanin, S. (2017). Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3571–3580.

[Duchi et al., 2018] Duchi, J. C., Jordan, M. I., and Wainwright, M. J. (2018). Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201.

[Dwork, 2006a] Dwork, C. (2006a). Differential privacy. In Bugliesi, M., Preneel, B., Sassone, V., and Wegener, I., editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Dwork, 2006b] Dwork, C. (2006b). Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer.

[ECML/PKDD, 2015] ECML/PKDD (2015). Taxi trajectory prediction dataset.

[Erlingsson et al., 2014] Erlingsson, Ú., Pihur, V., and Korolova, A. (2014). Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067. ACM.

[Esmerdag et al., 2016] Esmerdag, E., Gursoy, M. E., Inan, A., and Saygin, Y. (2016). Explode: An extensible platform for differentially private data analysis. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1300–1303. IEEE.

[Fanti et al., 2016] Fanti, G., Pihur, V., and Erlingsson, Ú. (2016). Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies*, 2016(3):41–61.

[Fiore et al., 2020] Fiore, M., Katsikouli, P., Zavou, E., Cunche, M., Fessant, F., Le Hello, D., Aivodji, U., Olivier, B., Quertier, T., and Stanica, R. (2020). Privacy

in trajectory micro-data publishing: a survey. *Transactions on Data Privacy*, 13:91–149.

[Garey et al., 1974] Garey, M. R., Johnson, D. S., and Stockmeyer, L. (1974). Some simplified np-complete problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, pages 47–63.

[Gu et al., 2019] Gu, X., Li, M., Cao, Y., and Xiong, L. (2019). Supporting both range queries and frequency estimation with local differential privacy. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 124–132. IEEE.

[Gu et al., 2020] Gu, X., Li, M., Cheng, Y., Xiong, L., and Cao, Y. (2020). Pckv: Locally differentially private correlated key-value data collection with optimized utility. In *2020 USENIX Security Symposium*.

[Gursoy et al., 2017] Gursoy, M. E., Inan, A., Nergiz, M. E., and Saygin, Y. (2017). Differentially private nearest neighbor classification. *Data Mining and Knowledge Discovery*, 31(5):1544–1575.

[Gursoy et al., 2018a] Gursoy, M. E., Liu, L., Truex, S., and Yu, L. (2018a). Differentially private and utility preserving publication of trajectory data. *IEEE Transactions on Mobile Computing*, 18(10):2315–2329.

[Gursoy et al., 2018b] Gursoy, M. E., Liu, L., Truex, S., Yu, L., and Wei, W. (2018b). Utility-aware synthesis of differentially private and attack-resilient location traces. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 196–211.

[Gursoy et al., 2020] Gursoy, M. E., Rajasekar, V., and Liu, L. (2020). Utility-optimized synthesis of differentially private location traces. In *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 30–39. IEEE.

[Gursoy et al., 2019] Gursoy, M. E., Tamersoy, A., Truex, S., Wei, W., and Liu, L. (2019). Secure and utility-aware data collection with condensed local differential privacy. *IEEE Transactions on Dependable and Secure Computing*.

[Hay et al., 2016] Hay, M., Machanavajjhala, A., Miklau, G., Chen, Y., and Zhang, D. (2016). Principled evaluation of differentially private algorithms using dpbench. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data*, pages 139–154. ACM.

[He et al., 2015] He, X., Cormode, G., Machanavajjhala, A., Procopiuc, C. M., and Srivastava, D. (2015). Dpt: Differentially private trajectory synthesis using hierarchical reference systems. *Proceedings of the VLDB Endowment*, 8(11):1154–1165.

[Ho and Ruan, 2011] Ho, S.-S. and Ruan, S. (2011). Differential privacy for location pattern mining. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '11, page 17–24, New York, NY, USA. Association for Computing Machinery.

[Hong et al., 2021] Hong, D., Jung, W., and Shim, K. (2021). Collecting geospatial data with local differential privacy for personalized services. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2237–2242. IEEE.

[Inan et al., 2016] Inan, A., Gursoy, M. E., Esmerdag, E., and Saygin, Y. (2016). Graph-based modelling of query sets for differential privacy. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, pages 1–10.

[Inan et al., 2020] Inan, A., Gursoy, M. E., and Saygin, Y. (2020). Sensitivity analysis for non-interactive differential privacy: Bounds and efficient algorithms. *IEEE Transactions on Dependable and Secure Computing*, 17(01):194–207.

[Johnson, 1985] Johnson, D. S. (1985). The np-completeness column: an ongoing guide. *Journal of Algorithms*, 6(3):434–451.

[Kairouz et al., 2014] Kairouz, P., Oh, S., and Viswanath, P. (2014). Extremal mechanisms for local differential privacy. In *Advances in Neural Information Processing Systems*, pages 2879–2887.

[Kim et al., 2018] Kim, J. W., Kim, D.-H., and Jang, B. (2018). Application of local differential privacy to collection of indoor positioning data. *IEEE Access*, 6:4276–4286.

[Leven and Galil, 1983] Leven, D. and Galil, Z. (1983). Np completeness of finding the chromatic index of regular graphs. *Journal of Algorithms*, 4(1):35–44.

[Li et al., 2020] Li, Z., Wang, T., Lopuhaä-Zwakenberg, M., Li, N., and Škoric, B. (2020). Estimating numerical distributions under local differential privacy. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 621–635.

[Ma et al., 2013] Ma, C. Y., Yau, D. K., Yip, N. K., and Rao, N. S. (2013). Privacy vulnerability of published anonymous mobility traces. *IEEE/ACM Transactions on Networking (TON)*, 21(3):720–733.

[Mir et al., 2013] Mir, D. J., Isaacman, S., Cáceres, R., Martonosi, M., and Wright, R. N. (2013). Dp-where: Differentially private modeling of human mobility. In *2013 IEEE International Conference on Big Data*, pages 580–588. IEEE.

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

[Qin et al., 2016] Qin, Z., Yang, Y., Yu, T., Khalil, I., Xiao, X., and Ren, K. (2016). Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 192–203. ACM.

[Ren et al., 2018] Ren, X., Yu, C.-M., Yu, W., Yang, S., Yang, X., McCann, J. A., and Philip, S. Y. (2018). Lopub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166.

[Shao et al., 2013] Shao, D., Jiang, K., Kister, T., Bressan, S., and Tan, K.-L. (2013). Publishing trajectory with differential privacy: A priori vs. a posteriori sampling mechanisms. In *International Conference on Database and Expert Systems Applications*, pages 357–365. Springer.

[Thakurta et al., 2017] Thakurta, A. G., Vyrros, A. H., Vaishampayan, U. S., Kapoor, G., Freudinger, J., Prakash, V. V., Legendre, A., and Duplinsky, S. (2017). Emoji frequency detection and deep link frequency. US Patent App. 15/640,266.

[Truex et al., 2020] Truex, S., Liu, L., Chow, K.-H., Gursoy, M. E., and Wei, W. (2020). Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*, pages 61–66.

[Wang et al., 2018a] Wang, N., Xiao, X., Yang, Y., Hoang, T. D., Shin, H., Shin, J., and Yu, G. (2018a). Privtrie: Effective frequent term discovery under local differential privacy. In *IEEE International Conference on Data Engineering (ICDE)*.

[Wang et al., 2019a] Wang, N., Xiao, X., Yang, Y., Zhao, J., Hui, S. C., Shin, H., Shin, J., and Yu, G. (2019a). Collecting and analyzing multidimensional data with local differential privacy. In *IEEE 35th International Conference on Data Engineering (ICDE)*, pages 638–649. IEEE.

[Wang et al., 2016] Wang, S., Huang, L., Wang, P., Nie, Y., Xu, H., Yang, W., Li, X.-Y., and Qiao, C. (2016). Mutual information optimally local private discrete distribution estimation. *arXiv preprint arXiv:1607.08025*.

[Wang et al., 2017] Wang, T., Blocki, J., Li, N., and Jha, S. (2017). Locally differentially private protocols for frequency estimation. In *Proc. of the 26th USENIX Security Symposium*, pages 729–745.

[Wang et al., 2019b] Wang, T., Ding, B., Zhou, J., Hong, C., Huang, Z., Li, N., and Jha, S. (2019b). Answering multi-dimensional analytical queries under local differential privacy. In *Proceedings of the 2019 International Conference on Management of Data*, pages 159–176.

[Wang et al., 2018b] Wang, T., Li, N., and Jha, S. (2018b). Locally differentially private frequent itemset mining. In *IEEE Symposium on Security and Privacy (SP)*. IEEE.

[Wang et al., 2019c] Wang, T., Li, N., and Jha, S. (2019c). Locally differentially private heavy hitter identification. *IEEE Transactions on Dependable and Secure Computing*.

[Welsh and Powell, 1967] Welsh, D. J. and Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86.

[Xiang et al., 2020] Xiang, Z., Ding, B., He, X., and Zhou, J. (2020). Linear and range counting under metric-based local differential privacy. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 908–913. IEEE.

[Yang et al., 2015] Yang, D., Zhang, D., Zheng, V. W., and Yu, Z. (2015). Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142.

[Yang et al., 2020] Yang, J., Wang, T., Li, N., Cheng, X., and Su, S. (2020). Answering multi-dimensional range queries under local differential privacy. *Proceedings of the VLDB Endowment*, 14(3):378–390.

[Ye et al., 2021] Ye, Q., Hu, H., Meng, X., Zheng, H., Huang, K., Fang, C., and Shi, J. (2021). Privkvm*: Revisiting key-value statistics estimation with local differential privacy. *IEEE Transactions on Dependable and Secure Computing*.

[Zhang et al., 2016] Zhang, J., Xiao, X., and Xie, X. (2016). Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data*, pages 155–170. ACM.

[Zhang et al., 2018] Zhang, Z., Wang, T., Li, N., He, S., and Chen, J. (2018). Calm: Consistent adaptive local marginal for marginal release under local differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 212–229. ACM.

[Zhao et al., 2020] Zhao, P., Zhang, G., Wan, S., Liu, G., and Umer, T. (2020). A survey of local differential privacy for securing internet of vehicles. *The Journal of Supercomputing*, 76(11):8391–8412.