Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Layla Pournajaf Date

Privacy-aware Task Management for Mobile Crowd Sensing

By

Layla Pournajaf
Doctor of Philosophy

Computer Science and Informatics

Li Xiong, Ph.D.
Advisor

Vaidy Sunderam, Ph.D.
Co-Advisor

Joyce Ho, Ph.D.
Committee Member

Cyrus Shahabi, Ph.D.
Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the Graduate School

Date

Privacy-aware Task Management for Mobile Crowd Sensing

By

Layla Pournajaf
Ph.D., Emory University, 2017

Advisor: Li Xiong, Ph.D.
Co-Advisor: Vaidy Sunderam, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2017

Abstract

Privacy-aware Task Management for Mobile Crowd Sensing
By Layla Pournajaf

Location-aware Mobile crowd sensing (MCS) has numerous applications in a wide range of domains including syndromic surveillance, crime mapping, traffic monitoring, and emergency response. Preserving the privacy of participants in such applications is one of the main challenges in developing effective task management solutions in MCS. Moreover, the inherent dynamic environment of MCS characterized by continuous change and inaccurate participant movement information pose further challenges for coordination of tasks and participants. Therefore, in this dissertation, we propose novel methods to build robust task management frameworks to handle uncertainty and ensure privacy in MCS applications. Our solutions not only increase the disposition of the participants to engage in data collection and sharing activity, but also ultimately lead to more effective MCS applications.

Privacy-aware Task Management for Mobile Crowd Sensing

By

Layla Pournajaf
Ph.D., Emory University, 2017

Advisor: Li Xiong, Ph.D.
Co-Advisor: Vaidy Sunderam, Ph.D.

A dissertation submitted to the Faculty of the Graduate School
of Emory University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2017

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my advisor, Dr. Li Xiong for all her support and guidance over the years. She has set an example of excellence as a researcher and mentor with her passion and dedication and I am very grateful for having her as my advisor. I would like to thank my co-advisor, Dr. Vaidy Sunderam for his absolutely invaluable advice, support, and constant encouragements through the course of my study. Many thanks to my committee members, Dr. Shahabi and Dr. Ho, for their feedbacks and precious insights during the course of my thesis. I would like to thank my fellow graduate students at AIMS, collaborators, and the staff at the Math&CS department for all of their help and support. Finally, I would like to thank my friends Farnaz and Rachel who were always there for me with their help and encouragements.

To my parents Reyhan and Ali who have always believed in me.
To my dear husband and friend Aras
who have always supported me with his advice, patience, and encouragements.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Mobile crowd sensing (MCS) [33] enables individuals to participate in a collective data sensing paradigm using their smartphones or other computing devices (i.e., contributing pictures, videos, audios, location, or speed measurements). Similar to the generic crowdsourcing models, MCS deals with a set of tasks (a.k.a jobs) which are completed by a set of participants (a.k.a workers). A general structure of task distribution in MCS is shown in Figure 1.1. Three main entities are identified as follows.

1. *Participants* are entities that use a sensing device to obtain or measure the required data about a subject of interest.

2. *Applications* or end users are the entities that request data through tasks and then utilize the information acquired by participants.

3. *Tasking entities* are responsible for storage and distribution of tasks to participants who meet the requirements of applications. In certain architectures, end users and participants can also act as tasking entities.

An interesting class of MCS (a.k.a. location-aware crowdsourcing) [4, 41, 50] deals with spatial tasks which are defined as specific targets including objects, events, or

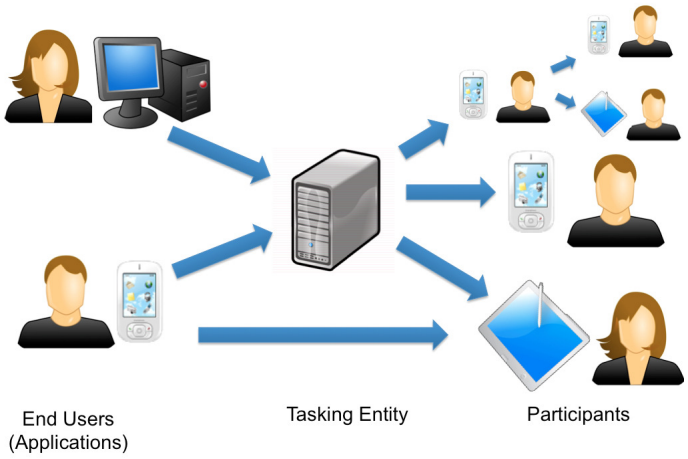


Figure 1.1: General structure of the task flow in MCS

phenomena at particular locations. Examples of spatial tasks include monitoring and reporting speed, traffic, or road conditions while driving (i.e. bumps, inclination, and elevation) [30, 42, 72]. Spatial tasks are considered *opportunistic* if they run in the background with little or no involvement from the participant (e.g. speed detection). In contrast, *participatory* tasks require the users to engage in data collection. For example, participants can report location-based conditions such as potholes or the quality of the road as they drive around in their normal commute [31, 90]. Participants may be also asked to search for the best prices located at different stores and report them to be shared with other users for finding the best prices in the region [15, 28].

As a crucial part of location-aware MCS, a task management framework is responsible for storing, indexing, and sharing geo-spatial tasks, recruiting qualified mobile participants, and acting as a coordinator between the users and applications. Task management frameworks in MCS fall into two major approaches: i) autonomous task selection, and ii) coordinated task assignment. In *autonomous task selection*, participants select their tasks autonomously from a set of existing tasks received from a task distribution entity. They might or might not inform the distributor about their

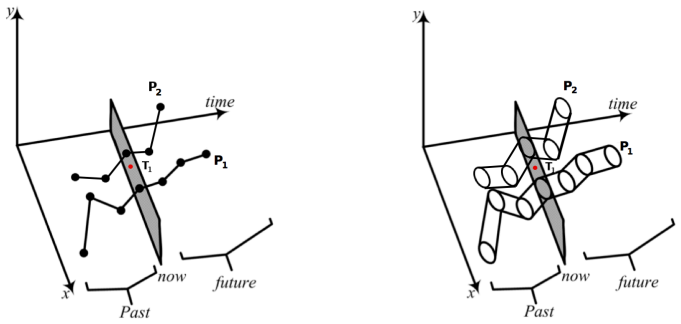
selection choices. Examples of these approaches may be found in [22, 27, 84]. *Coordinated task assignment* aims at optimizing the process of data sensing by efficient assessment of available sensing resources to meet the requirements of applications. The criteria for optimization of task assignment include sensing costs, coverage of targets of interest, quality, and credibility of sensed data. Examples of this approach can be found in [26, 50, 51, 79, 80, 85].

We identify two main challenges in spatial task management approaches in MCS. These challenges are i) *preserving the location privacy of participants*, and ii) *handling the noise and uncertainty of dynamic MCS environments*.

1.1.1 Participant Location Privacy

Participant location plays a crucial role in effective task management since tasks are assigned or selected by participants mainly based on their proximity to targets [50, 85]. In the presence of an untrustworthy task management server, privacy-aware participants may conceal their identity by anonymous contribution of data, however, their location may reveal their identity or other private attributes. Therefore, In many applications, participants might be reluctant to share their location with the task management framework due to privacy concerns. One promising approach to preserve location privacy is *spatial cloaking* which has been widely used in location-based services [6, 12, 34, 40]. MCS participants can adopt this method for location privacy, however, it introduces location uncertainty issues which the existing spatial task management approaches can not handle effectively. To overcome this problem, new *privacy-aware spatial task assignment* methods are necessary to exploit the cloaked locations of participants for efficient recruitment.

In addition to spatial cloaking, other notions of privacy such as *differential privacy* [29] and *private information retrieval (PIR)* [103] are also increasingly utilized in location-based services. In a recent work, differential privacy is adopted for spatial crowdsourcing task assignment [95] in which a trusted aggregator (e.g. a cell service provider) computes differentially private aggregated counts of participants in various



(a) Exact Trajectories (b) Uncertain Trajectories

Figure 1.2: Effect of uncertainty on spatial task assignment with two participants P_1 and P_2, and a task T_1.

spatial regions and provides them to tasking entities for task assignment. PIR-based methods have also been adopted for location-based services recently [34,35,53]. They guarantee cryptographic privacy by allowing data retrieval from a database without revealing any information to the database server about the retrieved item. This method is not utilized in spatial task management.

1.1.2 Dynamic MCS Environments

In addition to privacy concerns, the location of participants may become inaccurate as a result of error in location-detection devices or noisy transmissions. Moreover, participants may be constantly moving (e.g. commuters) and sensing tasks may be updated (e.g. moving crowd in an event) resulting in a dynamic and constantly changing environment.

Figure 1.2 illustrates an example of a spatial task assignment problem in dynamic environments. Two participant trajectories are indicated as P_1 and P_2 respectively, and the location of a spatial task is shown as T_1 at current time (i.e. labeled as now in the figure). Figure 1.2a includes exact trajectories as the sequence of locations and time stamps while Figure 1.2b only contains uncertain trajectories as the sequence of location areas at each time point. Assume that a task assignment

server has access to these trajectories, the location of the task, and an assignment goal (which requires tasks to be assigned to the closest participants at each time point). While it is intuitive to assign T_1 to P_1 at current time in Figure 1.2a using exact locations, it might not be as straightforward in Figure 1.2b, which deals with uncertain trajectories. Thus, there is an urgent need for dynamic task assignment in MCS for maximized sensing coverage and minimized sensing cost, while adapting in real-time to the dynamic and uncertain locations of participants and the sensing requirements of the applications.

1.2 Research Contributions

In this dissertation, we propose a novel framework for location-aware task management to handle uncertainty and ensure privacy in MCS applications while achieving desired task coverage with minimal cost. In Chapter 3 and Chapter 4, we present two spatial task assignment and selection methods to address the location privacy concerns of the participants in our framework. Chapter 5 and 6 extend the task management framework using an adaptive data-driven solution which deals with the continuous change and the uncertainty of dynamic environments in MCS. The rest of this section highlights the details of our contributions.

1.2.1 Privacy-aware Coordinated Spatial Task Assignment (Chapter 3)

In this chapter, we consider the spatial task assignment problem in a coordinated crowd sensing setting. We assume a tasking server is responsible for managing sensing tasks among participants who share their cloaked locations rather than their exact locations. Our goal is to efficiently assign sensing tasks to participants based on their cloaked locations to achieve a desired coverage objective with minimized cost (i.e. the total distance that participants have to travel for their assigned tasks). To

this end, we propose a novel approach for a privacy-aware coordinated spatial task assignment with the following contributions.

First, we propose a two-stage optimization approach for the spatial task assignment problem in the presence of cloaked locations. In the first stage, a global optimization problem is solved at the task assignment server using cloaked locations. Our approach addresses location uncertainty and can work with different spatial cloaking methods. In the second stage, participants individually fine-tune their assignments using their own exact locations. We formulate formal optimization objectives for each stage and further show the optimization problems at each stage are NP-hard. Second, we propose efficient greedy algorithms to solve the optimization problem at each stage. Finally, we implement our approach as a software and demonstrate, from both synthetic and real data, that our methods achieve high sensing coverage with low cost using cloaked participant locations.

1.2.2 Private Reverse K-Nearest Queries for Autonomous Spatial Task Selection (Chapter 4)

This chapter investigates a solution for an effective spatial task selection approach which guarantees strong location privacy for MCS participants. First, we propose a solution for reverse k nearest neighbor queries ($RkNN$) based on the concept of PIR which strongly preserves the privacy of the query point by preventing the location-based server from inferring any information about it. Then, we show that, this solution can be utilized in a privacy-aware autonomous spatial task selection approach for MCS to allow participants to retrieve tasks without compromising their location privacy.

Since the computational cost of PIR queries is high, it is important to propose solutions with minimum requirement of issuing PIR requests. While PIR based approaches have been proposed for kNN queries [77], designing PIR based solutions for $RkNN$ presents unique challenges as the number of $RkNN$ query answers can be arbitrary and often require searching over a large area for potential candidates.

We summarize our contributions below.

First, our solutions are optimized on the server-side by adopting proper data indexing models for efficient storage and effective data retrieval carefully designed for $RkNN$ queries. Second, clients are provided with a summary of the distribution of data in the form of aggregated counts. Such information when combined with the state-of-the-art uncertain $RkNN$ query processing methods, enable the clients to effectively prune the search space and compute the minimal necessary blocks of data for their answer. Finally, to guarantee near zero disclosure of information, we compute a query plan for each database which guarantees a fixed number of block retrievals from each database. This can answer any query regardless of its location, thus, the server does not learn any information about the query based on the number of requests or the size of the retrieved data.

1.2.3 Dynamic Spatial Task Assignment using Uncertain Trajectories (Chapter 5)

This chapter extends the coordinated task assignment problem for a more dynamic environment with moving participants. We propose a dynamic data-driven framework to deal with uncertain participant locations as a result of movement, noise, or error of location sensing infrastructure. Our approach is based on the DDDAS (Dynamic Data Driven Application Systems) [25] paradigm. The DDDAS concept is crucial to address the such crowd sensing applications in order to steer and assign the data collection tasks in targeted ways, adapting dynamically to application needs and the dynamic and uncertain locations of participants.

Our proposed task assignment framework entails a synergistic feedback loop between application simulations and data collection: 1) based on assigned tasks, participants report the collected data and possibly their current (uncertain) locations to the application; 2) the collected data are dynamically integrated into an executing simulation to augment or complement the application model (e.g. flood movement), 3) the reported (uncertain) locations are dynamically integrated into an executing

mobility model to accurately track participants' moving locations, and 4) conversely the executing simulations update the data collection targets and requirements as well as participants' locations which are then used by the task assignment module to make new task assignments and steer future data collection.

1.2.4 An Extensive Experimental Evaluation of Location Prediction Models For Moving participants (Chapter 6)

As a part of our dynamic framework for task assignment, this chapter carries out an extensive experimental evaluation for existing mobility modeling approaches. Several location prediction approaches are proposed in literature for variety of location-based services and social networks, however, it is still challenging to adopt the most suitable algorithms for specific applications. While complex algorithms are developed every year, yet simple Markov Chain algorithms has shown promising results [67] in location prediction and remain as the main solution to many real-world applications. Existing location prediction approaches target different applications with their parameters tuned for certain data sets. Many of such approaches have not been studied in variety of experimental settings with different data, thus, they lack an extensive evaluation to show their strengths and limitations in different situations. Moreover, due to existing of variety of spatial and temporal modeling of trajectories, several approaches are not comparable directly [73], therefore, building a framework to evaluate and compare such methods using the same data and test parameters remains as a challenge.

We perform an extensive evaluation of the existing state-of-the-art methods with the following contributions.

- We identify different location prediction problems in literature and discuss their similarities and differences. We also give a formal definition of the general next location prediction problem which is studied widely with applications in real-world location-based services.

- We categorize the existing approaches for next location prediction problem

based on several aspects including the level of personalization, the spatial and temporal representation of trajectories, and the mobility behavior modeling scheme.

- We study several state-of-the-art next location prediction algorithms and evaluate them extensively using both synthetic and real-world data sets. We design a series of experiments to evaluate and compare these algorithms based on the criteria of accuracy, efficiency, and robustness using data with different features.
- We use theoretical limits of human mobility prediction [88] to estimate the upper limits of predictability for our data sets. This feature allows us to identify the areas for improvement of algorithms in this study while comparing them to each other.

Chapter 2

Related Works

2.1 Spatial Task Management in Mobile Crowd Sensing

Spatial tasks require the participant to be at a specific place in order to fulfill a task. With the increasing use of smart phones with integrated GPS, the number of applications in which, tasks are assigned based on the location of participants has also grown. Monitoring and reporting speed, traffic, and road conditions are some examples of spatial tasks. Some of these tasks are opportunistic; they run in the background with little or no involvement from the participant which can be used to detect traffic speed, bumps, inclination, and elevation of the road [30, 42, 72]. In contrast, participatory tasks may ask the users to report potholes or the quality of the road as they drive around in their normal commute [31,90]. Spatial tasks are not restrained to reporting road conditions. For example, a participatory spatial task could require that the participants search for the best prices located at different stores and report them to provide other users with the best prices in the region [15, 28].

Spatial task management in mobile crowd sensing can be categorized into two major approaches: (i) Autonomous task selection, and (ii) Coordinated task assignment. In *autonomous task selection*, participants select their tasks autonomously from a set of existing tasks received from a task distribution entity. They might or might not

inform the distributor about their selection choices. Examples of these approaches can be found in [20–22, 27, 84]. Since the selected tasks in these methods are not optimized globally, these approaches might not be efficient with respect to sensing cost or global utility. Our proposed task selection approach in Chapter 4 seeks to overcome this shortcoming by utilizing $RkNN$ queries to increase the global task coverage while preserving privacy.

Coordinated task assignment aims at optimizing the process of data sensing by efficient assessment of available sensing resources to meet the requirements of applications. The criteria for optimization of task assignment include sensing costs, coverage of targets of interest, quality, and credibility of sensed data. Examples of this approach can be found in [26, 50, 51, 79, 80, 85]. Reddy et al. [79] proposed a coverage-based task assessment that finds the least costly subset of participants to achieve the coverage goal. Shirani-Mehr et al. [85] also proposed a coverage-based task assignment method for assigning viewpoints to a group of moving participants. Our proposed task assignment in Chapter 3 differs from these approaches since we use cloaked locations of participants for task assignments, thereby ensuring that the server does not learn the exact location of the participants.

2.2 Participant Privacy in Spatial Task Management

2.2.1 Location-based Privacy Attacks

Spatial tasks requested or accepted by participants might lead to disclosure of their current location and eventually their sensitive locations such as home/work addresses or even their identification through location-based attacks. Location-based attacks are widely recognized in the context of location-based services (LBS), however, certain attributes of mobile crowd sensing make it more vulnerable to some of spatial attacks. Here, we give a brief review of such attacks in MCS.

In frequent spatial tasks, even if the participant is using the application anonymously (e.g. using pseudonyms), her trajectory might reveal her sensitive locations or commutes [60] or eventually disclose her identity using location-based deanonymization attacks [32]. Krumm proposed several algorithms to identify a small group of anonymous participants and the home address of a larger group through location-based inference attacks [59]. They used the distribution of location traces during time, the last destinations of the day and the distribution of stay times to infer the home addresses of the participants. A location could be simply considered as home if it is visited frequently by the same user at night [16]. Participant locations can also be exploited using trajectory data mining algorithms [71] to identify their significant locations. The trajectory data can be also used to infer the individuals' life patterns (i.e. private schedules or lifestyles) [109]. Continuous or frequent spatial tasks make MCS more prone to location-based inference attacks as more location traces of participants are collected.

Kazemi et. al. [48, 49] defined a location-based attack in campaign-based Participatory Sensing applications when participants used Spatial k-anonymity [91] to hide their location. The location attack is defined as the identification of a participant by an untrusted server by learning the location of her issued task query. They observed that all participants of a campaign query spatial tasks from the server (a.k.a. all-inclusivity property) asking for tasks closer to them than any other participants (a.k.a range dependency property). These properties result in the server having spatially-dependent requests from all participants, so they argued that participatory sensing is more vulnerable to such location-based attack. Gonzalez et. al. showed that people and their movements are highly correlated [38] making such attacks possible.

Another location-based attack targets applications that utilize the density distribution of participants (i.e. aggregated number of participants) in a location for task management [86]. This attack exploited by a group of terrorists can be used to identify dense areas for explosive launches.

2.2.2 Spatio-Temporal Privacy-preserving Methods

With the growing advance of location-based services, several spatio-temporal privacy mechanisms have been developed recently (see recent surveys in [7,34,61]). Although the context in mobile crowd sensing is different from location-based services, these mechanisms can be used to address location privacy problems in such scenarios as well. Since location and time are two crucial pieces of information in an effective task management model, applying the existing spatio-temporal privacy-preserving techniques can be challenging. Here, we study some of the applicable methods in MCS task management.

Spatial Cloaking

In some crowd sensing applications, a perturbed or cloaked location can be used for spatial task management instead of exact locations. Spatial cloaking or perturbation hides the participant location inside a cloaked region using spatial transformations [52], generalization (e.g. k-anonymity) [46,101], or a set of dummy locations [54] in order to achieve location privacy. Some MCS applications do not require exact locations (e.g. pollution or weather monitoring), but for the majority of the applications with utility depending on location accuracy, adopting cloaking methods remains a challenge. In our proposed work in Chapter 3, participants share their cloaked location to obtain a set of closest tasks. We developed probabilistic methods to deal with uncertainty for a globally optimized task assignment.

Kazemi et. al. [48, 49] showed that spatial k-anoymity methods used in location-based services are not directly applicable to Participatory Sensing. Therefore, they proposed that a group of the representative participants ask for spatial tasks from an untrusted server, and share their results with the rest of participants. They would also adjust the spatial regions in queries to make queries independent from the location of other participants. Vu et. al. [97] proposed a spatial cloaking mechanism for Participatory Sensing based on k-anonymity and locality-sensitive hashing (LSH) to preserve both locality and k-anonymity.

While most traditional location cloaking methods rely on syntactic privacy models and are subjective to inference attacks, recent works applied more rigorous privacy notion based on differential privacy. The work in [6] proposed a location perturbation method based on a rigorous notion of indistinguishability, which is similar to the differential privacy concept. Another recent work [105] protects the exact locations with differential privacy in a proposed *delta*-location set, which is derived in Markov model to denote the possible locations where a user might appear at any time.

Temporally Constrained Sharing

Some approaches share exact locations for tasking, however, they avoid or mitigate the location based attacks to some extent by controlling the timing of disclosures. For example, to avoid frequent revealing of location of participants in spatial tasks, Krause et al. [58] use a spatial obfuscation approach. In their solution, they divide the space into a set of regions, then with a certain probability distribution, a subset of participants is selected in each region to report their exact location. Such methods can be used in traffic monitoring applications.

Another method [58] assigns spatial tasks to participants in a way that the number of tasks for each participant is minimized. In such an approach, there will be longer intervals between each location disclosure, mitigating location-based inference attacks. This scheme can be further controlled by participants by setting explicit policies regarding the intervals in which they prefer to share their location.

Aggregated Location with Differential Privacy

Differential Privacy [29] is a promising privacy-preserving approach with a strong protection guarantee. This method is adopted in privacy-preserving publishing of statistical information about location-based datasets [34] guaranteeing that individual location information disclosure does not occur. It can also prevent privacy attacks on the aggregated number of participants in a location as discussed in 2.2.1. In a recent work, differential privacy is adopted for spatial crowdsourcing task as-

signment [95] in which a trusted aggregator (e.g. a cell service provider) computes differentially private aggregated counts of participants in various spatial regions and provides them to tasking entities for task assignment.

Private Information Retrieval

Private information retrieval (PIR) protocol is a mechanism to ensure that location based services are unable to infer any information about user requests [35,53]. Among existing implementations of PIR protocol, hardware secure PIR accounts as the only practical PIR method [103]. This method relies on secure co-processor (SCOP) that is trusted by client and located in database servers. SCOP acts as a mediator between the client and the server by handling the clients' requests of data blocks, obliviously retrieving them from the databases on the server and returning the fetched data back to the client.

PIR approach has been recently used for answering private queries in Location-based services. Khoshgozaran et al. [53] proposed the use of SCOP to partially preserve the privacy of kNN queries. Since their approach allowed different number of PIR requests for different queries, it could lead to information leakage about the location of clients. Papadopoulos et al. [77] proposed a novel approach which ensures strong location privacy for kNN requests by assuming each query retrieves the same number of blocks from each database and the size of retrieved blocks are the same. This method is considered a practical technique that guarantees strong location privacy for kNN queries [77]. In Chapter 4, we adopt a similar approach to answer private $RkNN$ queries. However, given that $RkNN$ query is a more complex query type compared to kNN, private $RkNN$ queries introduce unique challenges. They not only require a different query answering approach, but also demand different indexing methods and query plan design. Moreover, the size of $RkNN$ query answers are not bounded like kNN queries (i.e. kNN queries always yield k answers) which makes it even more challenging. Other works have adapted this solution to other spatial queries such as shortest path queries [76], bichromatic

RNN (BRNN) [99] and kNN over road networks [98]. $BRNN$ [99] counts as the closest query type to our solution since it also handles reverse nearest neighbor queries, however, monochromatic queries are inherently different from bichromatic queries, so the Voronoi-based method which is used in their approach can not be adopted for them.

2.3 Reverse k-nearest neighbor (RkNN) queries

A considerable amount of research has been conducted to study different types of RkNN queries including snapshot RkNN queries [57], continuous RkNN queries [18], and probabilistic or uncertain RkNN queries [17]. In this section, we provide a brief overview of techniques that address RkNN queries in location-based services for both exact and uncertain data.

2.3.1 RkNN Queries in Exact Databases

RNN query was first introduced by Koren et al. [57]. They proposed a method to solve RNN query by pre-computation of NN circles for all data points. Techniques that do not utilize pre-computation are considered more effective in location-based services. Some of these techniques are six regions [89], TPL [93], [94], FINCH [104], VoR-tree [83], InfZone [18] and SLICE [107]. Most of these techniques utilize a pruning and verification framework to reduce the search space significantly. Pruning techniques are categorized into regions-based pruning [89,107] and half-space pruning methods [18, 93, 94, 104]. Stanoi et. al [89] answer RkNN queries by dividing the whole space centered at the query point q into six equal regions and find the kth nearest neighbor in each region. Any points that are located further from kth nearest neighbor of query point q in each region is pruned from the search space. Tao et. al [93, 94] propose TPL that utilizes the property of perpendicular bisectors to split the space between the query point q and any point p into two half-spaces. Any point located in the space that is contained by k half-spaces of other points is pruned.

2.3.2 RkNN Queries on Uncertain Data

The aforementioned approaches are not applicable to RkNN queries over uncertain data. Lian et al. [66] proposed a method based on spatial pruning to solve RkNN queries on probabilistic data objects. In their method, continuous probabilistic data are summarized into spheres which are used for pruning. Cheema et al. [17] designed a complex pruning method for uncertain RNN queries which consists of probabilistic pruning, half-space pruning, and dominance pruning followed by an optimized probabilistic verification method. Bernecker et al. [11] proposed an efficient pruning method on probabilistic data objects represented as discrete functions. Their approach utilizes two pruning strategies, spatial and probabilistic pruning. The spatial pruning phase uses Minimum distance-Maximum distance pruning and the probabilistic pruning phase applies a decomposition technique on uncertain data. Li et al. [65] developed a novel pruning algorithm for answering RkNN queries on uncertain data. They utilized two pruning strategies, spatial pruning in which pruned regions are computed by calculating the distance and the angular range between uncertain data, and the probabilistic method in which the upper bound for the probability of pruning more data is retrieved.

In Chapter 4, we adopt several such pruning approaches including Minimum distance-Maximum distance pruning method [11] and spatial pruning techniques based on half-spaces [17] and regions [65] for client side query processing based on the summary data (before retrieving exact data), in order to reduce the search space and minimize the PIR requests.

2.4 Participant Mobility Modeling and Prediction

Location is one of the most commonly used attribute in present databases. From a simple zipcode in U.S. Census data to GPS coordinates in navigation applications such as Google Maps, location data is collected everyday providing a rich source of information for a broad range of applications in transportation, epidemiology and

health-care, economic development, and marketing. An emerging family of such applications exploit the predictive property of human mobility. Location-based reminders/planners, cellular-user tracking [9], traffic prediction, epidemic prevention, and event prediction [114] are few examples of such applications. In Chapter 6, we present an extensive empirical evaluation of location prediction methods. To this end, we classify and review next location prediction approaches according to four different aspects: personalization, temporal representation, spatial representation, and mobility learning scheme.

Chapter 3

Spatial Task Assignment for Crowd Sensing with Cloaked Locations

In this chapter, we consider the spatial task assignment problem in a coordinated crowd sensing setting in which a tasking server is responsible for managing sensing tasks among participants who share their cloaked locations rather than their exact locations. Our goal is to efficiently assign sensing tasks to participants based on their cloaked locations to achieve a desired coverage goal with minimized cost, i.e. the total distance that participants have to travel for their assigned tasks.

3.1 Two-Stage Optimization Approach

In this section, we first define the spatial task assignment (STA) problem and then we formulate a version of STA which deals with cloaked locations (STAC) as a two-stage optimization problem.

3.1.1 Problem Definition

Figure 3.1 illustrates a high-level design for task management in a crowd sensing architecture. In our work, we focus on three main components of this architecture

including participants, applications and the tasking server. The applications are requesters of the data acquired via sensors carried/operated by participants. Our task management service referred to as the tasking server recruits suitable participants for applications. To this end, the applications upload their required tasks to the tasking service. A task includes a set of targets of interest and required sensing specifications such as type of sensing, required equipment, and sampling frequencies. Similarly, participants who are registered to this service via a trusted third-party anonymizer, provide their attributes including their capabilities such as their smart-device specifications, their spatial availability as cloaked areas, their temporal availability, and other restrictions such as their mobility limitations.

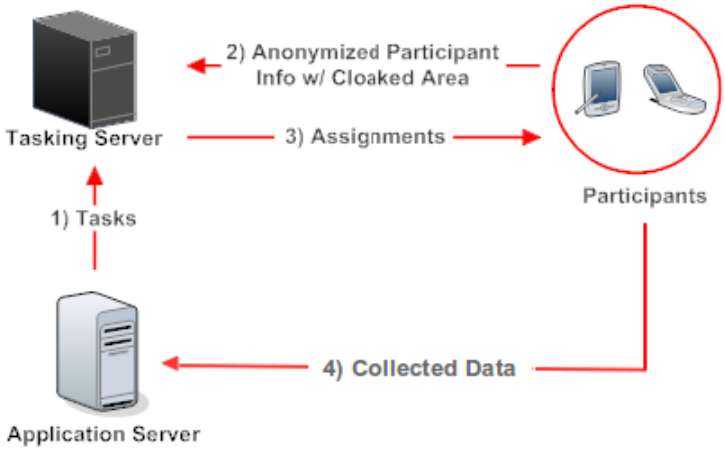


Figure 3.1: Task assignment in a crowd sensing architecture

In this section, we provide a more formal description of the spatial task assignment problem with cloaked locations. The summary of notations is presented in Table 3.1. First, we formally define, who is a participant, and what is a cloaked area.

Definition 3.1. *(Participant) A participant p_i is an anonymously registered user who has a limited travel budget b_i, i.e. the maximum distance a participant can travel. The participant shares this information as well as her cloaked area a_i (defined later in this section) and her desired sensing time with the tasking server. The participant's*

Table 3.1: Notations

p_i	Participant i
t_j	Target j
n	Number of participants
m	Number of targets
b_i	Travel distance budget of participant i
l_i	Location of the participant i
a_i	Cloaked area of participant i
k_j	Required coverage for target j
g	Required fraction of task coverage
$d_{i,j}$	Distance between the participant i and target j
$\mathbf{x}$	First stage assignment matrix
$\mathbf{y}$	Second stage assignment matrix
$\hat{\mathbf{d}}$	Expected distance matrix

true location l_i is considered private and is not shared with the server.

Definition 3.2. *(Cloaked Area) A cloaked area for a participant is a pair $\langle a_i, f_i \rangle$, where a_i is a spatial region and f_i is the probability density function of the participant at each point in a_i. For simplicity, we refer to the cloaked area as a_i in this chapter.*

Each participant is able to perform tasks that meet their restrictions. The following definitions describe what is a task, its assignment, and its coverage.

Definition 3.3. *(Task) A task specifies a set of targets for data collection, the location of each target, the required coverage for each target (k_j) (i.e. the number of participants to cover target t_j), and the overall coverage goal g (the required portion of the task coverage defined later in this section).*

Definition 3.4. *(Task Assignment) Task assignment is a mapping of participants to targets in a task shown by a matrix $\boldsymbol{x}$ where $x_{i,j} = 1$ if target $j \in M$ is assigned to*

participant $i \in N$, otherwise $x_{i,j} = 0$. $N := \{1, \ldots, n\}$ is a collection of row indexes (or participants), $M := \{1, \ldots, m\}$ is a collection of column indexes (or targets).

Definition 3.5. *(Task Coverage) Coverage for a target is defined as the number of participants assigned to it, normalized by the required coverage of the target k_j (e.g. for a target which requires two users to cover it $k_j = 2$, if only one user covers it, the target coverage would be 0.5). Task coverage (TU) is defined as the sum of coverage for all the targets in the task shown in (3.1). The maximum value of TU for full coverage is m. Coverage goal for a task denoted as g indicates the required fraction of task coverage with $g \in (0, 1]$.*

$$TU = \sum_{j \in M} \frac{\sum_{i \in N} x_{i,j}}{k_j} \tag{3.1}$$

Definition 3.6. *(Task Cost) The sensing cost for a pair of participant and target can be defined based on the travel distance, sensing duration or the complexity of each sensing. Since the participants may need to travel to the target location, we define a cost model which is simply the Euclidean distance between the participant's original location and assigned targets shown as a matrix $\boldsymbol{d}$. Task cost (TC) is defined as the sum of all sensing costs for all targets in the task. Our cost model can be substituted by any other distance-based cost model without affecting the problem definition.*

$$TC = \sum_{j \in M} \sum_{i \in N} x_{i,j} d_{i,j} \tag{3.2}$$

Given a set of participants and a task, we can define the task assignment problems as follows.

Definition 3.7. *(STA: Spatial Task Assignment) For a set of participants and the set of targets in a task, the spatial task assignment problem (STA) formulated below aims at achieving the task coverage goal with the minimum cost by assigning targets to qualified participants using their exact locations.*

$$\min_{x} \sum_{i \in N} \sum_{j \in M} d_{i,j} x_{i,j} \tag{3.3}$$

$$\text{s.t.} \quad \sum_{j \in M} \frac{\sum_{i \in N} x_{i,j}}{k_j} \geq gm$$

$$\forall_{i \in N} \sum_{j \in M} x_{i,j} d_{i,j} \leq b_i$$

where the minimization objective is to minimize the task cost TC, defined in (3.2). The first constraint indicates that the task coverage TU, defined in (3.1), has to be greater than or equal to the required task coverage gm. The second constraint represents the travel budget of each participant (i.e. the total travel distance for participant p_i can not exceed her travel budget b_i).

Definition 3.8. *(STAC: Spatial Task Assignment with Cloaked Locations) For a set of participants and the set of targets in a task, STAC aims at achieving the task coverage goal with minimum cost by assigning targets to the qualified participants using their cloaked locations. We formulate this problem as a two-stage optimization objective in Section 3.1.2.*

3.1.2 Formal Two-stage Optimization Objective

In spatial task assignment with cloaked locations (STAC), since exact locations of the participants are not provided to the server, the distance between targets and participants described by matrix $\mathbf{d}$, used as the sensing cost matrix, is unavailable to the tasking server. Therefore the server is required to deal with location uncertainty and estimate the values of $\mathbf{d}$ as an expected distance matrix $\hat{\mathbf{d}}$. Then, the server can utilize these expected values to perform the task assignment, however, this uncertainty introduces inaccuracy in distance estimations and subsequently in task assignments. Hence, we propose a two-stage optimization solution to solve STAC. The first stage optimization problem is a global task assignment problem (G-STAC)

based on uncertain locations solved at the tasking server, while the second is a local task assignment problem (L-STAC) solved by each participant. Dividing the assignment task into two separate problems utilizes participant location data locally while preserving participant privacy. The goal of the second stage is to refine and optimize task assignment results of the first stage by each participant using her exact location. In this section, we describe each stage in detail and then propose a formal objective for each problem.

G-STAC : First stage optimization objective

The first stage deals with uncertain locations which leads to uncertain distances for participant-target pairs. Assuming we had exact locations, the first-stage optimization objective would be as shown in (3). In absence of exact locations, we need to estimate distances as $\hat{\mathbf{d}}$. We discuss the estimation process with more details in Section 3.2.1.

L-STAC : Second stage optimization objective

Our second stage optimization runs in the participant's device locally using the given assignment from the first stage. Since new information is introduced in the second stage (i.e., exact locations available in each participant's device), these assignments can be adjusted and refined for more coverage and/or lower distance/cost. The reason is that (i) some targets might have been assigned to the participant by the server based on the estimated distance, but they are not actually accessible to the participant as the exact distance may exceed her travel budget; (ii) some targets are very close to the participant but have been estimated as being farther and not assigned. If each participant simply selects the closest targets in the second stage, over-coverage may occur for some of the targets meaning they might be covered more than required. Therefore, in addition to minimize the total travel distance with the exact location in the second stage, we would like to keep the assignments of the first stage unchanged as much as possible because they have been globally optimized

for the global coverage goal and cost. The objectives of second stage assignment optimization of each participant $p_i, i \in N$ is shown in (3.4).

$$\begin{aligned} & \min_{y} \sum_{j \in M} d_{i,j} y_{i,j} \\ \text{s.t.} \quad & |\mathbf{y}_i - \mathbf{x}_i| < \epsilon \\ & \sum_{j \in M} \frac{y_{i,j}}{k_j} \geq \sum_{j \in M} \frac{x_{i,j}}{k_j} \\ & \sum_{j \in M} y_{i,j} d_{i,j} \leq b_i \end{aligned} \tag{3.4}$$

where for each participant p_i, $\mathbf{x}_i$ is the first stage assignment vector, $\mathbf{y}_i$ is the second stage assignment vector, $\mathbf{d}_i$ is the distance vector, b_i is the participant's travel budget, $|\mathbf{y}_i - \mathbf{x}_i|$ is the Hamming distance between two binary vectors $\mathbf{x}_i$ and $\mathbf{y}_i$ which is constrained using a small threshold ϵ in favor of keeping the first-stage assignments unchanged as much as possible. The second constraint guarantees that p_i's contribution to the task coverage is at least equal to the coverage share assigned to her in the first stage. The last constraint guarantees that her travel distance is within her budget.

3.1.3 Complexity Analysis

In this section we show that our global and local problems are NP-hard, by reducing the minimum set cover problem to G-STAC and L-STAC. The minimum set cover problem is a well studied NP-hard problem defined as follows.

Definition 3.9. *(Minimum Set Cover Problem [96]) Given a universe W, a collection S of subsets of W, and a cost function $c : S \to \mathbb{R}_+$ find a minimum cost sub-collection of S that covers each element of W.*

Theorem 3.10. *The G-STAC is an NP-hard optimization problem.*

Proof. To prove that G-STAC is NP-hard we show a polynomial reduction of the minimum set cover problem (Definition 3.9) to our problem.

Consider a minimum set cover problem with $W = \{p_1, \ldots, p_n, p_\emptyset, t_1, \ldots, t_m, t_\emptyset\}$ and S being a set of two-element subsets of W, i.e., $S = \{\{p_i, t_j\} : p_i \in W, t_j \in W\}$. Let $k > 0$ and $c : S \to \mathbb{R}_+$ be a cost function such that $c(\{p_i, t_j\}) = \hat{d}_{i,j}$ ($p_i \neq p_\emptyset$ and $t_j \neq t_\emptyset$) is an expected distance between t_j and p_i. For remaining elements of S the cost function is defined as follows: $c(\{p_i, t_\emptyset\}) = 0$ and $c(\{p_\emptyset, t_j\}) = D$, where $t_j \neq t_\emptyset$ and $D > \sum_{i \in N, j \in M} c(\{p_i, t_j\})$.

We reduce such instance of the minimal set cover problem to the G-STAC problem as follows. Let $P = \{p_i : i = 1, \ldots, n\}$ be a set of participants and $T = \{t_j : j = 1, \ldots, m\}$ be a set of targets. A distance between t_j and p_i is equal to $d_{i,j} = c(\{p_i, t_j\})$. We assume G-STAC has an optimal solution $\mathbf{x}_{OPT}$ with minimum cost and full coverage (when setting $g = 100\%$ and $k = 1$). We derive $S_{OPT} \subset S$ from $\mathbf{x}_{OPT}$ for the minimal partial set cover problem as follows. If t_j is assigned to p_i in $\mathbf{x}_{OPT}$, i.e. $x_{i,j} = 1$, then $\{p_i, t_j\} \in S_{OPT}$. If t_j is not assigned to any participant in $\mathbf{x}_{OPT}$, then $\{p_\emptyset, t_j\} \in S_{OPT}$. If p_i has no target assigned to it in $\mathbf{x}_{OPT}$, then $\{p_i, t_\emptyset\} \in S_{OPT}$. If all targets have been assigned and each participant has at least one target assigned to it in $\mathbf{x}_{OPT}$, then $\{p_\emptyset, t_\emptyset\} \in S_{OPT}$.

We show by contradiction that S_{OPT} covers set W with the minimal cost, i.e., any other solution would not have lower cost. Let us assume by contradiction that there is S' that covers W with lower cost. Elements of S' can be mapped to assignment pairs of participants to targets, therefore they define a solution $\mathbf{x}'$ for the G-STAC problem. Since S has a lower cost than S_{OPT}, then $\mathbf{x}'$ has a lower cost than $\mathbf{x}_{OPT}$. This is a contradiction with $\mathbf{x}_{OPT}$ being the optimal solution of G-STAC. □

Similarly, we can show that L-STAC is NP-hard by a polynomial reduction of the minimum set cover problem (Definition 3.9) to it.

3.2 Algorithms

In this section, we propose efficient greedy algorithms to approximate the optimization objectives for both G-STAC and L-STAC.

3.2.1 First Stage: G-STAC

We first present two methods to deal with location uncertainty in the first stage, then we propose an efficient greedy algorithm to approximate the optimization objective for G-STAC based on the greedy solution proposed in [87] for partial set cover problem.

Distance Estimation

As mentioned earlier, we use a distance-based cost model in our work which defines the sensing cost as the Euclidean distance between participants and targets. Therefore, our tasking server is required to deal with the location uncertainty of the participants to estimate distances. Queries over uncertain spatio-temporal data have been extensively studied with many algorithms to handle queries such as nearest neighbors, top-k, and range queries [102]. Most of these methods aim at ranking the query answers, thus cannot be directly adopted in our work which requires actual distances to optimize the sensing cost. Having the cloaked areas (as the pair of the area and the probabilistic density function $\langle a, f \rangle$), we propose two simple methods to calculate the expected distances.

i) Centroid-point: In this baseline method, we calculate the centroid of all points in the cloaked area $z \in a$ as the expected location of the participant and use it to calculate the expected distances $\hat{\mathbf{d}_{i,j}}$.

$$\hat{d}_{i,j} = dist(\int_{z \in a_i} z f_i(z) dz, l_j)$$

where l_j is the location of the target j and the function $dist$ is the Euclidean distance between two points.

ii) Expected-probabilistic: In this method, for each pair $\langle i, j \rangle$ of participant-target, we first calculate the probability of the target j being accessible by the participant i as $\rho_{i,j}$ (i.e., the probability that target j is within the travel budget of the

participant i). To calculate this probability, we apply a simple pruning approach for each participant-target pair and shrinks the cloaked area a_i to a'_i which is the area of intersection between a_i and the circle centered at target j with radius b_i (i.e. the distance budget of participant i). Then, having the probability density function f_i, we calculate the probability of the participant being in a'_i which is equal to the probability of the target j being within the travel budget of participant i ($\rho_{i,j}$).

$$\rho_{i,j} = \int_{z \in a'_i} f_i(z) dz$$

Finally, we compute $\hat{d}_{i,j}$ as the expected distance between the target and the intersection area a'_i with probability $\rho_{i,j}$.

$$\hat{d}_{i,j} = \frac{\int_{z \in a'_i} dist(z, l_j) f_i(z) dz}{\int_{z \in a'_i} f_i(z) dz}$$

The above estimation methods can work with any cloaking area or can be discretized to work with a set of perturbed locations associated with probabilities. Figure 3.2 illustrates our estimation approaches when participant location is cloaked in a circular region with uniform probability distribution function.

Greedy Algorithm

Algorithm 1 represents the pseudocode for an efficient greedy algorithm to approximate the solution of our first stage objective. It iteratively picks the most cost-effective participant-target pair and updates the current coverage for the target, until either the coverage goal is met or all travel budgets of participants are exhausted. Since both the number of targets to be assigned and all travel budgets do not increase in time and always have non-negative values, the number of updates is finite. In each iteration, the algorithm finds the most cost-effective participant-target pair and assigns them to each other. For a participant $p_i, i \in N$ and target $t_j, j \in M$, the cost-effectiveness of assigning them to each-other is calculated as $\phi^{(1)}_{i,j}$ which is the ratio of expected distance $\hat{d}_{i,j}$ (cost) to the expected coverage contributed by this

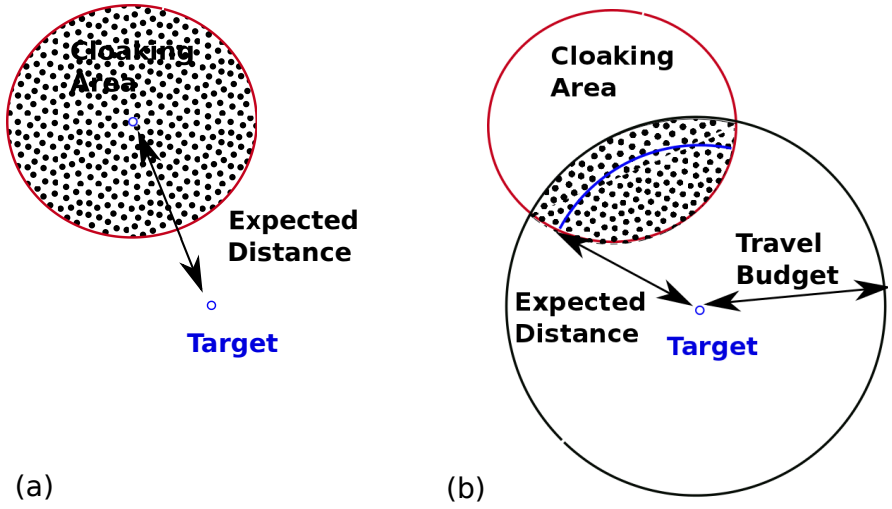


Figure 3.2: (a) Centroid-point method, (b) Expected-probabilistic method

participant.

$$\phi_{i,j}^{(1)} = \frac{\hat{d_{i,j}}}{\min(1 - u_j^+, \frac{1}{k_j}) + \epsilon}$$

$\mathbf{u}^+$ is the vector of current covered portions of the targets which is initially all zero. If a target is fully covered, the corresponding value of this target in $\mathbf{u}^+$ becomes 1. The expected coverage contributed by participant p_i for target t_j is hence $\min(1 - u_j^+, \frac{1}{k_j})$, the minimum of remaining required coverage of t_j and the coverage p_i can offer for t_j. Finding the minimum aims at preventing over-coverage. The small positive value ϵ is added to avoid overflow when the expected coverage by the participant is zero.

Since one of our distance estimation methods is probabilistic, Algorithm 1 is designed to select the most cost-effective pair of participant-target $\langle i, j \rangle$ with probability $\rho_{i,j}$. For the Centroid-point method, these probabilities are calculated as

$$\rho_{i,j} = \begin{cases} 1 & \hat{d}_{i,j} \leq b_i \\ 0 & \hat{d}_{i,j} > b_i \end{cases}$$

For the probabilistic method, $\rho_{i,j}$ is calculated as described in Section 3.2.1. Al-

gorithm 1 finds the answer in one pass through all participant-target pairs for the centroid-point method because the probabilities are either 0 or 1. For the expected-probabilistic method an upper-bound threshold R is used in the algorithm to stop the algorithm after R passes through all possible pairs. While the algorithm will converge after sufficient number of passes, we use R mainly for experiment purposes and enhanced efficiency. At the end of the first stage, the covered proportion of targets is updated in $\mathbf{u}^+$ based on the first stage assignments. Therefore, we refer to it as the expected coverage vector which is passed to the participant in the second stage along with her first stage assignment and the set of her accessible targets.

Time Complexity. The time complexity of our distance estimation methods are $O(nms)$ where n is the number of participants, m is the number of targets, and s is the number of points (sampling points when f_i is continuous) in each participant's cloaked area. Algorithm 1 runs in $O(nm)$ for the centroid-point method because the probabilities are either 0 or 1, so the algorithm finds the answer in one pass through all participant-target pairs. For the expected-probabilistic approach, the algorithm will run no more than R rounds for all participant-target pairs, so the the time complexity is $O(Rnm)$.

3.2.2 Second Stage: L-STAC

Due to the uncertainty of locations used in the first stage, a participant might be assigned to targets that are not accessible while not being assigned to targets that are accessible. The main goal of the second stage is for each participant to fine-tune the assignment using her exact location while maintaining the overall coverage goal. The main pitfall in the second stage is over-coverage of some targets at the cost of under-coverage of others, i.e. more participants are assigned to some targets than required, if each participant simply optimizes its cost by selecting the closest targets in the second stage. Hence, we have an additional constraint to bound the overall changes in the assignments implemented by several heuristics in order to maintain the overall coverage goal without incurring additional cost. The server provides the

Algorithm 1 A greedy algorithm for the first stage task assignment problem

Input: P (set of participants), T (set of targets), **b** (vector of travel budgets), $\hat{d}$ (matrix of expected distances), k (vector of required coverage for targets), g (task coverage goal), ρ (matrix of the access probabilities), R (threshold on running rounds)

Output: **x** (matrix of task assignments), $\mathbf{u}^+$ (vector of covered portion of targets)

All elements of **x** and $\mathbf{u}^+$ are initialized to 0.
$TC \longleftarrow 0$
$r \longleftarrow 0$
while $(TC \leq gm)$ **and** $(r < R)$ **do**
  if a remaining probable pair exists **then**
    Select the most cost-effective target-participant pair from the remaining pairs, say indexed at i and j with the probability $\rho_{i,j}$.
    if a pair is selected **then**
      Assign the selected pair as $x_{i,j} \longleftarrow 1$
      $TC \longleftarrow \min(1 - u_j^+, \frac{1}{k_j}) + TC$
      $u_j^+ \longleftarrow \min(1 - u_j^+, \frac{1}{k_j}) + u_j^+$
      $b_i \longleftarrow b_i - \hat{d}_{i,j}$
      if $u_j^+ = 1$ **then**
        $T \longleftarrow T \setminus T_j$
      end if
      if $b_i = 0$ **then**
        $P \longleftarrow P \setminus P_i$
      end if
    else
      $r \longleftarrow r + 1$
    end if
  else
    Break
  end if
end while

expected coverage vector, the final $\mathbf{u}^+$ at the end of G-STAC to all participants. Hence, the $\mathbf{u}^+$ at the beginning of second stage optimization is initialized with the given values from the server.

Algorithm 2 presents the pseudo code for our greedy approach to approximate the solution of our second stage objective. This algorithm runs locally on each participant's device $p_i \in P$, so it has access only to the corresponding participant's attributes including her exact location, and the information provided by the server including the set of candidate targets τ that may be accessible by the participant (the server can prune the targets that are not accessible by the participant if the minimum possible distance between a participant and a target is larger than b_i), and the result of the first stage assignment for this participant $\mathbf{x}_i$. The result of assignments in this algorithm is stored in $\mathbf{y}_i$.

The algorithm at participant p_i first initializes all elements of its assignment vector to 0 (no assignment) and updates $\mathbf{u}^+$ so it only contains the coverage contributed by all other participants, i.e. by removing the current targets assigned to p_i from the first stage (line 1-8). Similar to Algorithm 1, the algorithm then iteratively picks the most cost-effective target and assigns it to p_i with some probability which is designed to avoid over-coverage. In contrast to Algorithm 1, since we want to satisfy the the first constraint of (3.4), we penalize each new assignment which is different from $x_{i,j}$. Therefore, the cost-effectiveness score of each assignment in this stage is calculated as $\phi^{(2)}_{i,j}$.

$$\phi^{(2)}_{i,j} = \frac{\frac{d_{i,j}}{b_i} + |x_{i,j} - 1|}{\min(1 - u^+_j, \frac{1}{k_j}) + \epsilon}$$

which is the ratio of second stage cost (i.e., the sum of normalized distance and change penalty) to the expected coverage contributed by this participant for target $t_j \in \tau$. The expected coverage contributed by the participant is computed the same as in the first stage. The other difference of our second stage algorithm from the first stage concerns the probabilities which are used to assign targets to participants. For a target j, $\rho_{i,j}$ is calculated as follows,

$$\rho_{i,j} = 1 - \frac{\phi^{(2)}_{i,j}}{\max\left\{\phi_{i,j}{}^{(2)}\right\}}$$

Using this probability, we aim at avoiding over-coverage of the targets, but at the

same time reducing the chances of costly assignments. Without this probability, participants would repeatedly assign targets until their travel budget is exhausted. Completely expending the travel budget by all participants can result in over-coverage with high cost. This effect can be seen easily in the autonomous task selection methods discussed in Chapter 2. Using $\phi^{(2)}$ to calculate this probability favors more cost-effective assignments by giving them a higher probability. Similar to Algorithm 1, for efficiency purposes, an upper-bound threshold R' is used in the algorithm to stop the algorithm after R' passes through all targets in τ.

Time Complexity. Algorithm 2 runs no more than R' rounds of passing through all targets in τ, so having the number of all targets as m, the time complexity is $O(R'm)$.

Algorithm 2 A greedy algorithm for the second stage task assignment problem

Input: $p_i,\ i \in N$ (the participant), τ (set of accessible targets for p_i), $\mathbf{x}_i$ (first stage assignments for p_i), b_i (p_i's travel budget), $\mathbf{k}$ (vector of required coverage values for targets), g the task coverage goal, $\mathbf{u}^+$ (vector of covered portion of targets), R' (threshold on running rounds)

Output: $\mathbf{y}_i$ (vector of task assignments for p_i)

All elements of $\mathbf{y}_i$ are initialized to 0.
$LC \longleftarrow 0$ (Local task coverage). $AC \longleftarrow 0$(Assigned task coverage to this participant)
$r \longleftarrow 0$
for all the targets in τ **do**
    $u_j^+ \longleftarrow u_j^+ - \frac{x_{i,j}}{k_j}$
    $AC \longleftarrow AC + \frac{x_{i,j}}{k_j}$
end for
while $(LC < AC)$ **and** $(b_i > 0)$ **and** $(r < R')$ **do**
    if a remaining probable target exists in τ **then**
        if target j is selected **then**
            if $d_{i,j} \leq b_i$ **then**
                Assign the selected target as $y_{i,j} \longleftarrow 1$
                $TC \longleftarrow TC + \min(1 - u_j^+, \frac{1}{k_j})$
                $u_j^+ \longleftarrow u_j^+ + \min(1 - u_j^+, \frac{1}{k_j})$
                $b_i \longleftarrow b_i - d_{i,j}$
                $\tau \longleftarrow \tau \setminus \tau_j$
            end if
        else
            $r \longleftarrow r + 1$
        end if
    else
        Break
    end if
end while

3.3 Experimental Results

In this section, we evaluate our task assignment methods experimentally using both real and synthetic datasets. First we discuss the details of our experiment settings, then we present and analyze the results.

3.3.1 Settings

Datasets. We used a real dataset from Gowalla [1] which contains check-in information of users of a location-based social network. The check-ins consist of time and location coordinates of users at different positions. For our experiments, we used user and position information during October 2010 in New York city. We used each day as a snapshot for our task assignment experiment. In all experiments, participants are selected uniformly from all Gowalla users on each given day, while targets are picked randomly from all the spots.

We also used Brinkhoff's Network-based Generator of Moving Objects [13] to create a set of synthetic dataset of moving objects (OLE) to test our methods and algorithms. The map of the city of Oldenburg in Germany is used as the input to the generator. In OLE, at each time snapshot, the set of participants is chosen uniformly from the set of moving objects in the map. In the same way, targets are selected from the nodes of the road graph of the map.

Evaluation Metrics. Task cost (TC) and task coverage (TU) are calculated as described in Section 3.1. In many settings, the desired coverage goal (g) may not be achievable, even given exact locations of the participants, due to the limited number of participants or travel budgets of the participants. Hence, we also present a combined cost metric that adds the task cost (TC) and uncovered targets ($gm-TU$) and normalizes the sum to the range of [0,1] using min-max method. We refer to this normalized value as penalized cost (PC):

$$PC = \frac{(gm - TU) + TC}{gm + \sum_i b_i}$$

The denominator is used for normalization and is equal to the maximum possible value for the sum of uncovered portion of the task and the task cost. Thus, a smaller value of PC represents higher coverage and lower cost, which is considered a better result. In our experiments, we study the effect of different parameters such as the number of participants/targets, coverage goal, and cloaking size on the task cost and coverage.

Parameters. Table 3.2 shows the parameter settings for our simulations with default settings highlighted. In all experiments, we selected the travel budget of the participants for walking randomly in the range of 1%-3% of the map size. We experimented with different cloaking models such as circular and rectangular areas, continuous/discrete instance sampling, and uniform/normal probability distribution of the instances, however, since the results were very similar for different cloaking models, we only present results for a circular model with continuous uniform distribution of instances. The size of the cloaking area for each participant is roughly selected uniformly in the range of 0.01%-1% of the map area. For all of our experiments with the OLE dataset, we used a part of the map of Oldenburg highlighted in Figure 3.3. The reason of this selection is to provide a more crowded map, compared to Gowalla which is a sparse dataset with a maximum of 235 participants in each snapshot. The required coverage of each target (k) is set to one in all experiments because varying the required target coverage has a similar effect as varying number of targets. The coverage goal g is selected between 10%-100% of the number of targets, with a default value of 100%. R and R' vary depending on the size of the cloaking area, but in general they are selected in the range of [50,200]. We repeated each experiment for 100 times and obtained their average as our final results.

Comparisons. We report the results of the following methods based on the combination of different distance estimation model of the G-STAC (Centroid-point or Expected-probabilistic) and the optimization stages (one-stage G-STAC-only or two-stage G-STAC/L-STAC combination).

- CPA1 (baseline): one-stage centroid-point based approach as a baseline,
- CPA2: two-stage centroid-point based approach to demonstrate the benefit of the two-stage optimization compared to the baseline one-stage optimization approach,
- EPA1: one-stage expected-distance approach to demonstrate the benefit of probabilistic distance estimation over the baseline centroid approach,

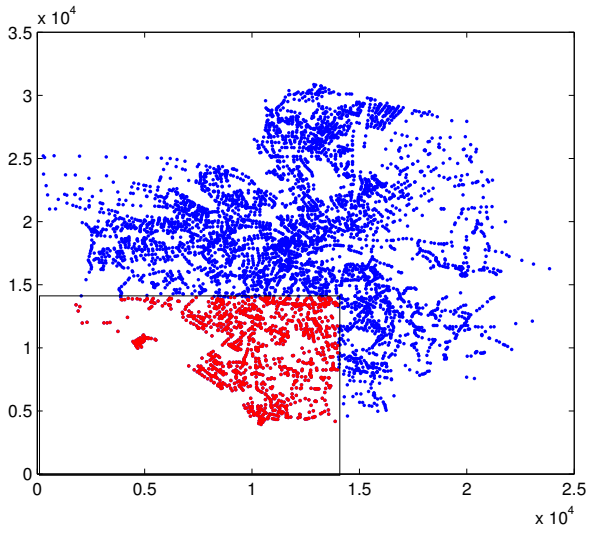


Figure 3.3: The map of Oldenburg, Germany generated by [13]. The employed section is framed.

- EPA2 (complete proposed solution): two-stage expected-probabilistic approach,
- NPA (reference solution): we utilized our first stage optimization solution with zero level of privacy as a reference solution with no privacy constraint (NPA). In NPA, we assume the tasking server has access to exact locations of the participants, therefore it runs only on the server.

Table 3.2: Experimental settings with highlighted default values

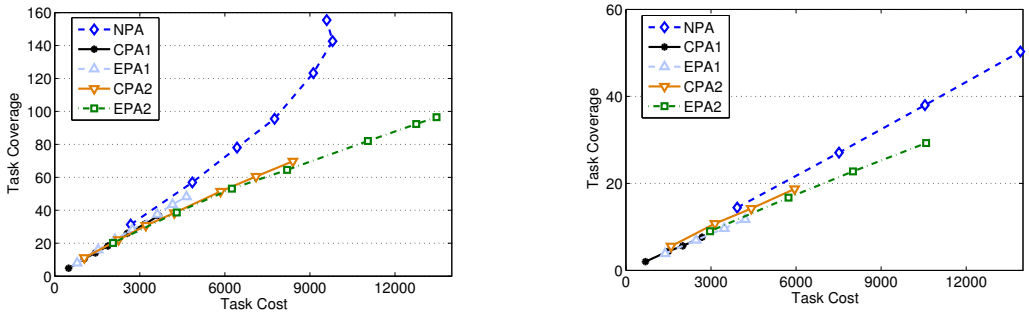
Parameter	**Value**
Number of Participants	50, 100, 150, **200**, 300, 400, 500
Number of Targets	50, 100, 150, **200**, 300, 400, 500
Travel Budget	1% - 3% of Map Size
Coverage Goal	10%-**100%**
Cloaking Model	**Circular**, Rectangle
Cloaking Area	0.01% - 1% of Map Area

3.3.2 Results

In this section, we report the results of each experiment for the two datasets in terms of task cost, coverage, and penalized cost. The scales for the task coverage and cost are different for the two datasets due to their different map sizes and level of sparsity.

Impact of Numbers of Participants and Targets

In these experiments, we study the impact of increasing the number of participants and targets on the task cost and coverage by: (a) varying the number of participants while the number of targets is fixed; and (b) varying the number of targets while the number of participants is fixed.



(a)

(b)

Figure 3.4: Task coverage vs cost for different number of participants, and $m = 200$ using datasets (a) OLE (b) Gowalla

Figure 3.4 shows the task coverage versus cost in both datasets for increasing number of participants with a fixed number of targets using the default settings. Overlapping points in some approaches such as baseline indicate that increasing the number of participants does not affect the task cost or coverage in some cases. In both datasets, our two-stage approaches (CPA2 and EPA2) achieve a significantly higher coverage compared to the one-stage approaches (CPA1 and EPA1). Thanks to the local optimization at the second stage, they are able to get much closer to the coverage goal while the ratio of cost and coverage stays roughly the same. Moreover,

the expected-probabilistic approaches outperform their corresponding centroid-point methods which is more apparent for CPA2 and EPA2. In OLE, EPA2 achieves more coverage with the same coverage/cost ratio as CPA2, but in Gowalla, this ratio is higher for CPA2, which means by using EPA2, more coverage is obtained at the expense of slightly higher cost per additional coverage due to the sparsity of the dataset.

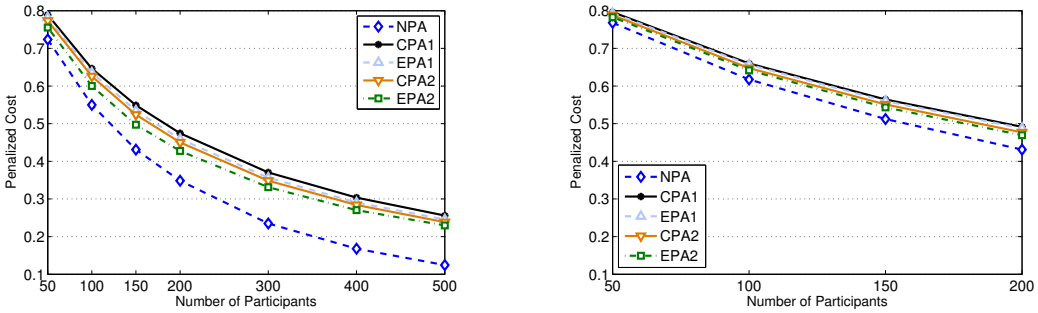
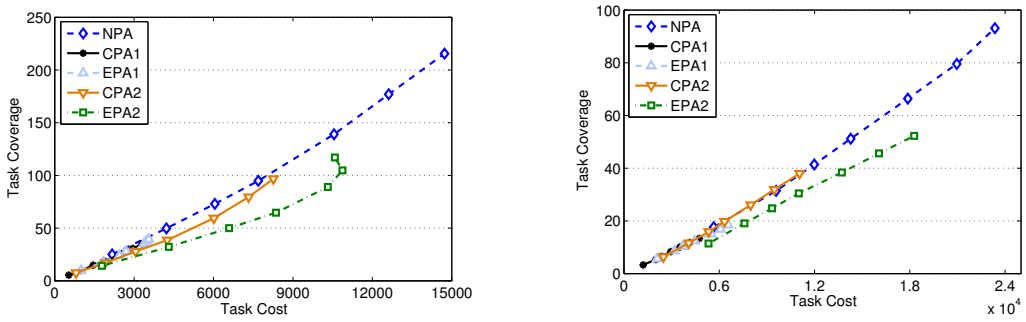


Figure 3.5: Penalized cost for different number of participants, and $m = 200$ using datasets (a) OLE (b) Gowalla

Figure 3.5 shows the penalized cost in both datasets for increasing number of participants with a fixed number of targets using the default settings. Increasing the number of participants results in higher task coverage which causes lower penalized costs for all of the approaches. In both datasets, for all combinations of the participants and targets, the expected-probabilistic approaches outperform their corresponding centroid-point approaches. This result is more clear in the OLE dataset due to higher task coverage being possible. On the other hand, regardless of the distance estimation methods, both two-stage methods outperform the one-stage methods.

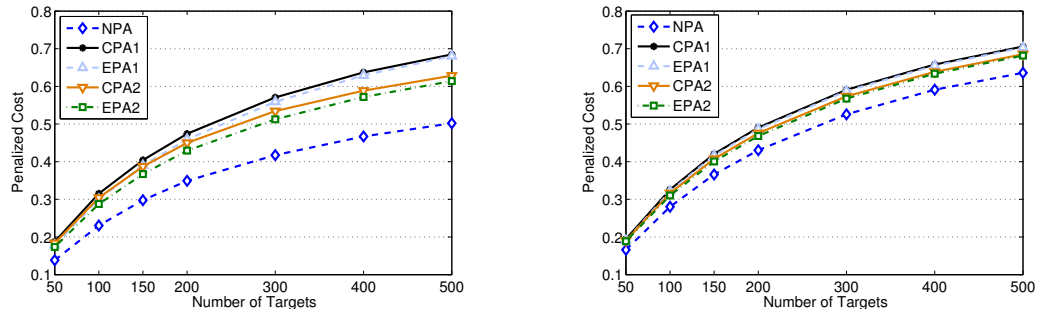
Figure 3.6 shows task coverage versus cost in both datasets for increasing numbers of targets with a fixed number of participants using the default settings. Overlapping points in some approaches such as baseline indicate that increasing the number of targets does not affect the task cost or coverage in some cases. In both datasets,



(a) (b)

Figure 3.6: Task coverage vs cost for different number of targets, and $n = 200$ using datasets (a) OLE (b) Gowalla

our two-stage approaches (CPA2 and EPA2) achieve higher coverage compared to the one-stage approaches (CPA1 and EPA1). Moreover, for the same number of targets, the expected-probabilistic approaches achieve higher coverage compared to their corresponding centroid-point methods. All methods are robust preserving a constant coverage/cost ratio, however, while CPA2 keeps a ratio comparable to NPA, EPA2 achieves more coverage at the expense of higher cost per additional coverage.



(a) (b)

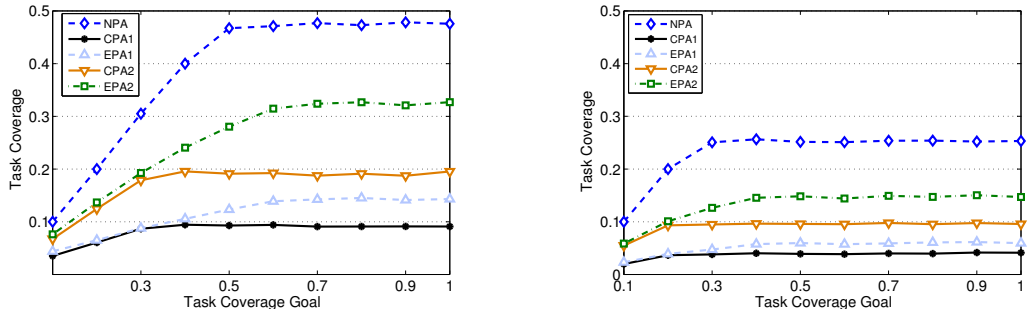
Figure 3.7: Penalized cost for different number of targets, and $n = 200$ using datasets (a) OLE (b) Gowalla

Figure 3.7 shows the penalized cost in both datasets for increasing numbers of

targets with a fixed number of participants using the default settings. In both datasets, for all combinations of participants and targets, the expected-probabilistic approaches outperform their corresponding centroid-point approaches and similarly, both two-stage methods outperform the one-stage methods. For the same experiment settings in OLE and Gowalla, the difference between penalized cost of different methods including NPA is smaller in Gowalla.

Impact of Coverage Goal

Figure 3.8 shows task coverage in both datasets for increasing coverage goal with a fixed number of participants and targets using the default settings including 200 participants and 200 targets. In both datasets, our two-stage approaches (CPA2 and EPA2) achieve higher coverage compared to the one-stage approaches (CPA1 and EPA1). Moreover, for the same coverage goal, the expected-probabilistic approaches achieve higher coverage compared to their corresponding centroid-point methods. Comparing the real dataset Gowalla to synthetic OLE, all methods achieve higher coverage in OLE which can be explained by higher density and lower sparseness of participants.



(a) (b)

Figure 3.8: Relative task coverage for different coverage goal (relative), $n = 200$, and $m = 200$ using datasets (a) OLE (b) Gowalla

Figure 3.9 shows task coverage versus cost for the same experiment setting. In

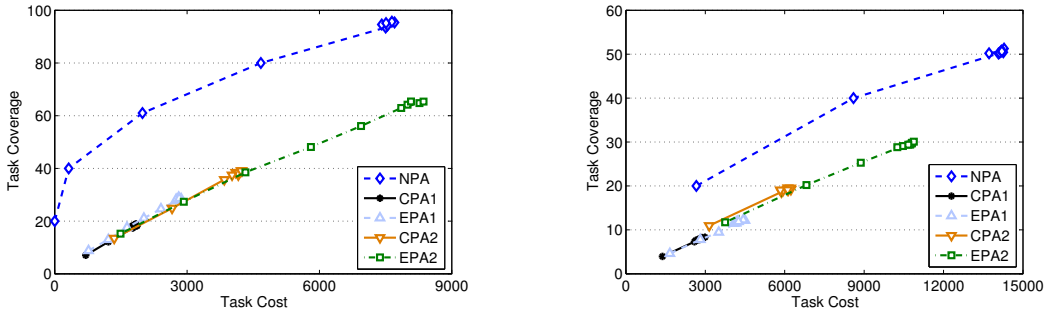


Figure 3.9: Task coverage vs cost for different coverage goal (relative), $n = 200$, and $m = 200$ using datasets (a) OLE (b) Gowalla

both datasets, EPA2 achieves a higher task coverage for the same coverage goals, with a coverage/cost ratio very similar to other methods. Overlapping points in each approach indicate that changing the coverage goal does not affect the task cost or coverage in some cases.

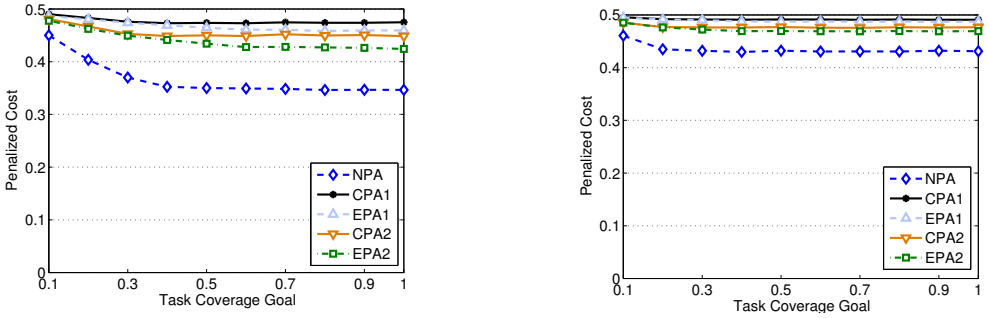
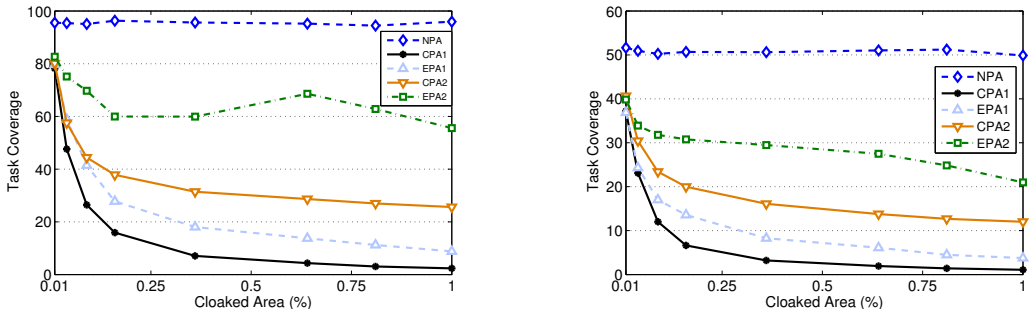


Figure 3.10: Penalized cost for different coverage goal (relative), $n = 200$, and $m = 200$ using datasets (a) OLE (b) Gowalla

Finally, Figure 3.10 shows the impact of coverage goal on penalized cost. EPA2 outperforms the other methods for all values of coverage goal.

Impact of Cloaking Size

Figure 3.11 shows the impact of cloaking size on task coverage for a fixed number of participants and targets. The cloaking size is shown as a percentage of the map area. By increasing the cloaked size, in both datasets, EPA2 shows more robustness compared to CPA2 as well as the one-stage methods, indicating that EPA2 is not affected by cloaking size as much as the other methods. Figure 3.12 shows the impact of cloaking size on cost and coverage for the same experiment. Similarly, in both datasets, CPA1, EPA1, and CPA2 are more affected by cloaking size. Overlapping points in some approaches such as NPA indicate that changing the cloaking size does not affect the task cost or coverage in some cases. Finally, Figure 3.13 shows the impact of cloaking size on penalized cost. EPA2 outperforms the other methods for all cloaking sizes.



(a) (b)

Figure 3.11: Task coverage for different sizes of cloaking area, $n = 200$, and $m = 200$ using datasets (a) OLE (b) Gowalla

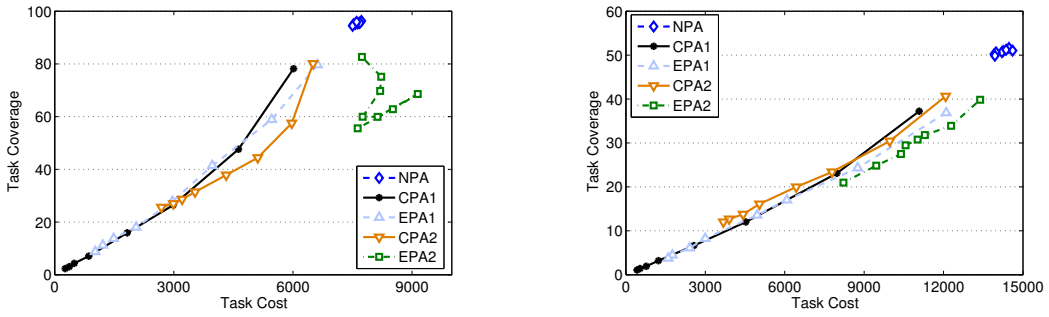


Figure 3.12: Task coverage vs cost for different sizes of cloaking area, $n = 200$, and $m = 200$ using datasets (a) OLE (b) Gowalla

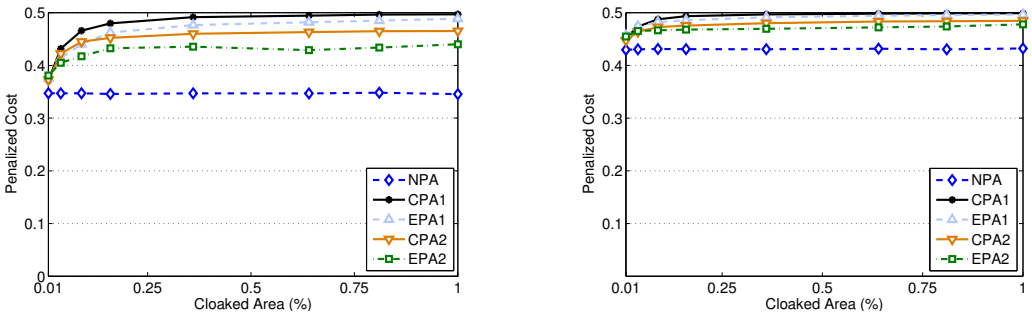


Figure 3.13: Penalized cost for different sizes of cloaking area, $n = 200$, and $m = 200$ using datasets (a) OLE (b) Gowalla

Chapter 4

Private Reverse K-Nearest Queries for Autonomous Spatial Task Selection

In this chapter, we investigate a solution based on the notion of private information retrieval (PIR) to be utilized in an *autonomous spatial task selection* method in which participants retrieve tasks without compromising their location privacy. This approach differs from our privacy-aware spatial task assignment solution discussed in Chapter 3 since i) it follows a task selection paradigm which does not require coordination, thus, tasks are retrieved autonomously by participants from a task server, and ii) participants do not share any information with the server as they utilize PIR for retrieving tasks without the server learning which tasks are accessed and downloaded.

To this end, first, we propose a solution for answering private reverse k-nearest neighbor queries (RkNN) based on the concept of PIR which strongly preserves the privacy of the query point by preventing the location-based server from inferring any information about it. Then, we show that this method can be utilized in autonomous task selection in MCS.

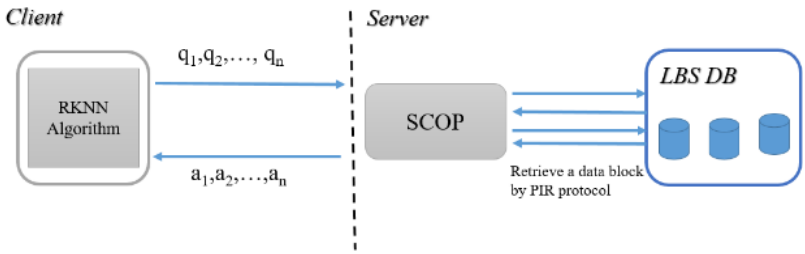


Figure 4.1: System model

4.1 Problem Definition: Private RkNN Queries

In this section, we provide an overview of the system model and formally define our problem. Our high-level system model is illustrated in Figure 4.1. We utilize secure co-processor PIR protocol (SCOP) [103] to answer private RkNN queries. PIR co-processor is installed in the server and is trusted by the client while the server is not trusted. We define RkNN query and the problem of private RkNN query answering as below.

Definition 4.1. *For a set of data points P and a query point q, RkNN query retrieves every points $p \in P$ for which Dist(q,p) $\leq$ Dist(p,p_k) where p_k is the kth nearest point to p in $P - \{q\}$.*

Definition 4.2. *For any query point q, Private RkNN Query Answering ensures $RkNN_q$ is computed in a manner that an untrusted data service provider does not learn about the query answer and cannot distinguish between the query point and any other point.*

In our model, a private RkNN query answer is retrieved by the client through several rounds of accessing different database layers. In each round the client issues a set of block retrieval requests through SCOP. SCOP uses PIR protocol to obliviously access blocks of information and fetch the requested blocks of data from the requested layer in the server. Then, SCOP returns the fetched data to the client. This data is used by client to partially answer the query and also determine the required data from

the next layer. This procedure repeats for every data layer until RkNN query is completely answered. The number of rounds is determined by the number of data layers and the size of PIR requests from each layer is pre-computed by the server as a *query plan* when databases are built.

Definition 4.3. *For a set of database layers DB_1, DB_2,...,DB_n, a query plan is a tuple $qp =< cnt_1, cnt_2, ...cnt_n >$ which indicates the number of data blocks that should be retrieved from each layer respectively in the given order.*

One major requirement of the query plan is that the order of access to the database layers and the number of PIR requests to each layer should be the same for any query point. This guarantees the server does not learn any information about the query based on the number of requests or the size of the retrieved data. This is a key attribute of our model that should be followed strictly. Due to this requirement, the query plan is pre-computed by the server and shared with the client before she issues a query.

We propose two solutions RkNN-HG and RkNN-HRT for answering private RkNN queries through PIR. Our solutions involve three major parts; 1) Server-side data storage and indexing, 2) Client-side query processing, and 3) Query plan pre-computation. Our solutions are discussed in detail in the next sections.

4.2 Server-side Indexing

In this section, first, we describe a naive approach for indexing data using the methods proposed in [77]. This approach is based on a fixed grid structure for indexing and storing data points in the server. Then, we introduce two improved indexing methods RkNN-HG and RkNN-HRT which optimize query answer retrieval through PIR for RkNN queries. In Section 4.3, we propose a client-side query processing method which is applicable to both of our indexing methods with some modifications.

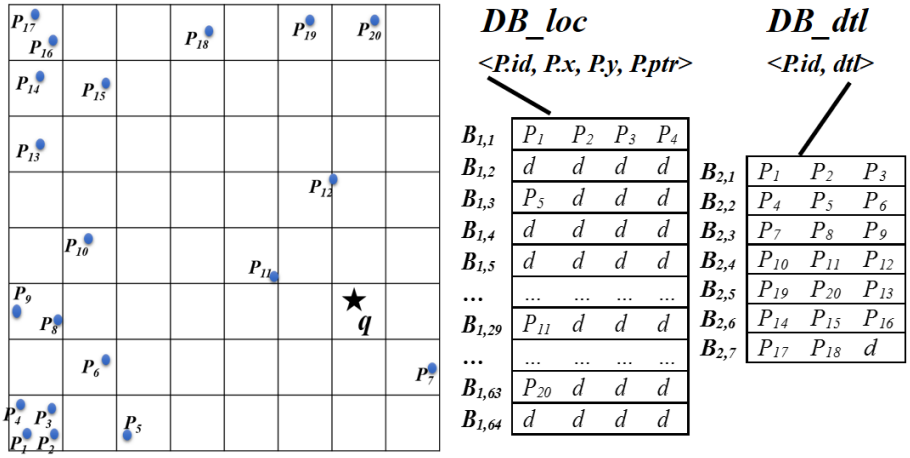


Figure 4.2: DB structure of naive solution

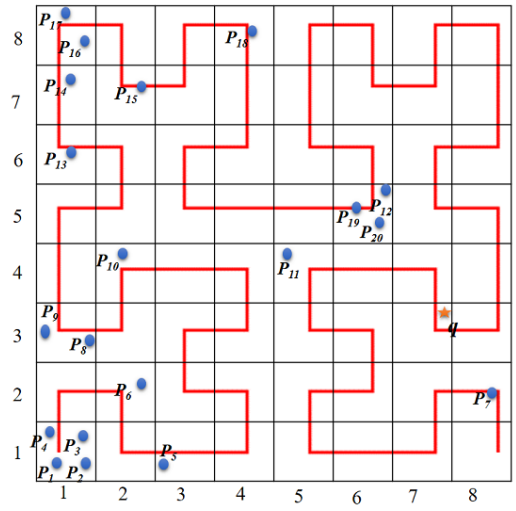
4.2.1 Naive Indexing

Figure 4.2 shows the server-side database structure for a sample dataset. For indexing the data points, a grid G is built and stored as two databases DB_{loc} and DB_{dtl}. DB_{loc} stores the coordinates of data points by allocating one disk block per grid cell c. For example, the points in cell $c_{1,1}$ are stored in block $B_{1,1}$. The coordinates of each point p is stored in the form of $< p.id, p.x, p.y, p.ptr >$ where $p.id$ indicates the unique identifier of p, $p.x$ and $p.y$ are the location coordinates of it, and $p.ptr$ indicates a pointer to DB_{dtl} where the rest of the information about p is stored.

Since the points in one cell may underflow or overflow a block of disk space, we use the following strategies. In the case of an underflow, we fill the empty space with dummy points shown as d. For overflow, we allocate a new block of data and store a pointer to it at the end of the overflown block. The second database, DB_{dtl} stores the extra information associated with each data point in the form of $< p.id, p.detail >$.

This approach has two weaknesses that should be addressed. Since it needs to store the information of each cell in one block, a skewed distribution of data would result in a larger database with wasted space. Furthermore, retrieving dummy data through expensive PIR requests can affect the run time of queries significantly. Therefore,

we propose a more efficient indexing method RkNN-HG.



(a)

DB_cnt

C_{11}	C_{12}	C_{22}	C_{21}	C_{31}	C_{41}	C_{42}	C_{32}
0,4	4,0	4,1	5,0	5,1	6,0	6,0	6,0
C_{33}	C_{43}	C_{44}	C_{34}	C_{24}	C_{23}	C_{13}	C_{14}
6,0	6,0	6,0	6,0	6,1	7,0	7,2	9,0
C_{15}	C_{25}	C_{26}	C_{16}	C_{17}	C_{18}	C_{28}	C_{27}
9,0	9,0	9,1	9,1	10,1	11,2	13,0	13,1
C_{37}	C_{38}	C_{48}	C_{47}	C_{46}	C_{36}	C_{35}	C_{45}
14,0	14,0	14,1	15,0	15,0	15,0	15,0	15,0
C_{55}	C_{65}	C_{66}	C_{56}	C_{57}	C_{58}	C_{68}	C_{67}
15,0	15,3	18,0	18,0	18,0	18,0	18,0	18,0
C_{77}	C_{78}	C_{88}	C_{87}	C_{86}	C_{76}	C_{75}	C_{85}
18,0	18,0	18,0	18,0	18,0	18,0	18,0	18,0
C_{84}	C_{83}	C_{73}	C_{74}	C_{64}	C_{54}	C_{53}	C_{63}
18,0	18,0	18,0	18,0	18,0	18,1	19,0	19,0
C_{62}	C_{52}	C_{51}	C_{61}	C_{71}	C_{72}	C_{82}	C_{81}
19,0	19,0	19,0	19,0	19,0	19,0	19,1	20,0

DB_loc

<P.id, P.x, P.y, P.ptr>

$B_{2,1}$	P_1	P_2	P_3	P_{11}
$B_{2,2}$	P_7	P_8	P_5	P_6
$B_{2,3}$	P_{18}	P_{19}	P_{20}	P_{17}
$B_{2,4}$	P_{15}	P_{16}	P_{14}	P_{13}
$B_{2,5}$	P_{10}	P_{12}	P_9	P_4

DB_dtl

<P.id, dtl>

$B_{3,1}$	P_1	P_2	P_3
$B_{3,2}$	P_{11}	P_7	P_8
$B_{3,3}$	P_5	P_6	P_{18}
$B_{3,4}$	P_{19}	P_{20}	P_{17}
$B_{3,5}$	P_{15}	P_{16}	P_{14}
$B_{3,6}$	P_{13}	P_{10}	P_{12}
$B_{3,7}$	P_9	P_4	d

(b)

Figure 4.3: DB structures in RkNN-HG (a) fixed grid with Hilbert-Curve filling, and (b) databases

4.2.2 RkNN-HG: Hilbert Grid Indexing

For more efficient data storage and retrieval, the fixed regular grid structure can be improved by incorporating Hilbert curve filling method as proposed in [77] to solve private kNN queries. In this section, we adopt their indexing approach on the server-side in our RkNN-HG method. For every grid cell, we compute a Hilbert score which is then used to determine the index of the block to store its enclosed data points and their coordinates. Using Hilbert curve assures that the locality of data is preserved

(i.e. data points in adjacent cells are more likely to be stored in the same or adjacent blocks of database). It also eliminates the need for storing dummy data, therefore, it saves space significantly.

Figure 4.3 depicts a Hilbert curve with granularity 8×8 for indexing an example database DB into three database layers DB_{cnt}, DB_{loc} and DB_{dtl} constructed by the server. DB_{cnt} records a tuple $< c.pre, c.count >$ for every cell c in the grid in the order of Hilbert space-filling curve [82]. These parameters, $c.pre$ and $c.count$, respectively indicate the number of preceding data points and the number of points in the cell c. The intuition behind computing this aggregated meta-data about each cell is to provide clients with enough information to prune unnecessary cells before retrieving required data using PIR. In our experiments, we observed that unlike kNN, a RkNN query may require the aggregated number of data points in every grid cell in the map to be able to compute its answer, therefore, we share DB_{cnt} with clients and do not store them in the databases that are accessed through PIR.

The second database DB_{loc} stores the location coordinates of all points using the same order as DB_{cnt} for each data point. This means DB_{loc} stores a tuple $< p.id, p.x, p.y, p.ptr >$ for every point p where $p.id$ is the unique identifier of p, $< p.x, p.y >$ represents the location coordination of p, and $p.ptr$ refers to the pointer to access the actual data in DB_{dtl}. Finally, DB_{dtl} stores detailed information of each data point in which each entry has $< p.id, p.detail >$ which represents additional data associated with it.

4.2.3 RkNN-HRT: Hilbert R-Tree Indexing

The RkNN-HG technique described in the previous section stores the aggregated counts of data for every cells of the fixed grid structure. The RkNN query processing algorithm (as we will explain in Section 4.3.2) examines each cell against RkNN candidates and uses pruning to avoid retrieval of data points from the cells that do not contain RkNN results. Therefore, for a high resolution grid structure, the number of cells increases resulting in a larger DB_{cnt} data and higher computational

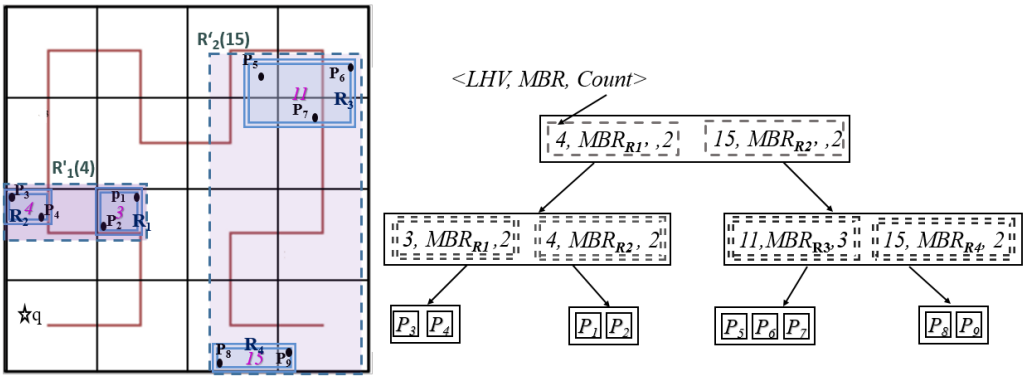


Figure 4.4: Structure of Hilbert R-tree.

cost during pruning phase. To overcome this problem, we propose a new method using Hilbert R-tree [47]. Kamel et al. [47] proposed Hilbert R-tree indexing method which is a hybrid structure based on B^+-tree and R-tree. It uses Hilbert curve to keep the locality of spatial data points. By adopting an object-based indexing using minimum bounding rectangles rather than a space-based method using fixed grid cells, we overcome the problems associated with non-uniform distributions of data including ineffective pruning of sparse regions. Moreover, it may reduce the DB_{cnt} data size that needs to be shipped to the client.

Figure 4.4 illustrates the Hilbert R-tree for our sample data points. Internal nodes of Hilbert R-tree consist of entries in the form of $< lhv, mbr, count, ptr >$ where lhv refers to the largest Hilbert value of all corresponding children of the node, mbr indicates the minimum bounding rectangle (MBR) that covers all of the points in the corresponding sub-tree, $count$ is the number of data points enclosed in the mbr, and ptr presents the pointer to the next level. Hilbert R-tree uses the largest Hilbert value of the MBR to determine internal node splitting instead of considering the area or the distance between MBRs like R-tree. Entries in the leaf nodes are in the form of $< mbr, pid >$, where mbr represents the MBR of the point and pid indicates the leaf identifier. In addition to keeping the locality of points by using Hilbert-based B^+-tree, another important feature of Hilbert R-tree is that nodes at each tree level

are also ordered based on their Hilbert values; consequently, points in the leaves are stored in the order of their Hilbert values too.

We follow the same database design for storing data as RkNN-HG method with minor modifications. After building the Hilbert R-tree structure for a dataset, DB_{cnt} is constructed as a set of tuples including the metadata summarizing the information of leaf nodes. For every leaf l a tuple is stored as $< l.mbr, l.lhv, l.count >$, where, $l.mbr$ represents the boundary of the rectangle that includes all of the data points of the corresponding leaf, $l.lhv$ is the largest Hilbert curve value of the data points in the leaf, and $l.count$ is the number of points enclosed at the leaf MBR. DB_{loc} and DB_{dtl} are built in the same manner as RkNN-HG method.

4.3 Client-side Query Processing

Given the server-side indexing of the data, users issue RkNN queries to the SCOP using the information provided by the server including the indexing parameters such as grid granularity g or hilbert resolution h, and a fixed query plan qp. The query plan is a tuple $qp =< cnt_{loc}, cnt_{dtl} >$ which is pre-computed in the server and indicates the number of blocks that should be retrieved from DB_{loc} and DB_{dtl} respectively. We describe the required pre-computations in Section 4.4. The reason for using a query plan is that regardless of the location of client, she should be able to answer her RkNN query using a fixed number of data blocks. In this way, queries cannot be differentiated by DB server based on the number or size of retrieved blocks. In order to guarantee that every user receives enough blocks of data for an accurate answer, cnt_{loc}, and cnt_{dtl} constitute upper bounds for the number of blocks needed by any possible query point. In addition to the query plan, clients are provided with DB_{cnt}, so they are able to prune unwanted cells before retrieving actual coordinates from DB_{loc}. The size of DB_{cnt} is data-independent and only varies by g while both DB_{loc} and DB_{dtl} are data-dependent.

Algorithm 3 RkNN query answer retrieval algorithm

Input: q (query location coordinate), k (parameter of $RkNN$ query), qp (query plan), g (grid granularity), DB_{cnt} (aggregated count data), map (map dimensions)

Output: $RkNN_q$ (RkNN answer set)

1: $RkNN_q$, CND_c (set of candidate cells) and CND_p (set of candidate points) initialized to $\emptyset$
2: $Q \longleftarrow$ locateQueryCell(q,map,g)
3: **if** $Q.count > 0$ **then**
4: Add Q to CND_c
5: $Q.INF_c \longleftarrow$ findInfluenceCells(Q)
6: **end if**
7: **while not** {all cells in a layer around Q are pruned k times} **do**
8: **for** every unvisited cell c in the closest layer of cells around Q **do**
9: $c.visited \longleftarrow 1$
10: **for** every cell cnd in CND_c **do**
11: **if** prune(cnd,c) **then**
12: $c.pruned \longleftarrow c.pruned + cnd.count$
13: **end if**
14: **end for**
15: **if** $c.pruned < k$ **then**
16: Add c to CND_c
17: $c.INF_c \longleftarrow$ findInfluenceCells(c)
18: **end if**
19: **end for**
20: **end while**
21: **for** every cell cnd in CND_c **do**
22: **if** prunedByInfluenceCells(cnd) **then**
23: remove cnd from CND_c
24: **end if**
25: **end for**
26: $CND_p, INF_p \longleftarrow$ retrievePIRData(DB_{loc}, CND_c, INF_c)
27: **for** every point cnd in CND_p **do**
28: **if** prunedByInfluencePoints(cnd) **then**
29: remove cnd from CND_p
30: **end if**
31: **end for**
32: $RkNN_q \longleftarrow$ retrievePIRData(DB_{dtl}, cnd_points)

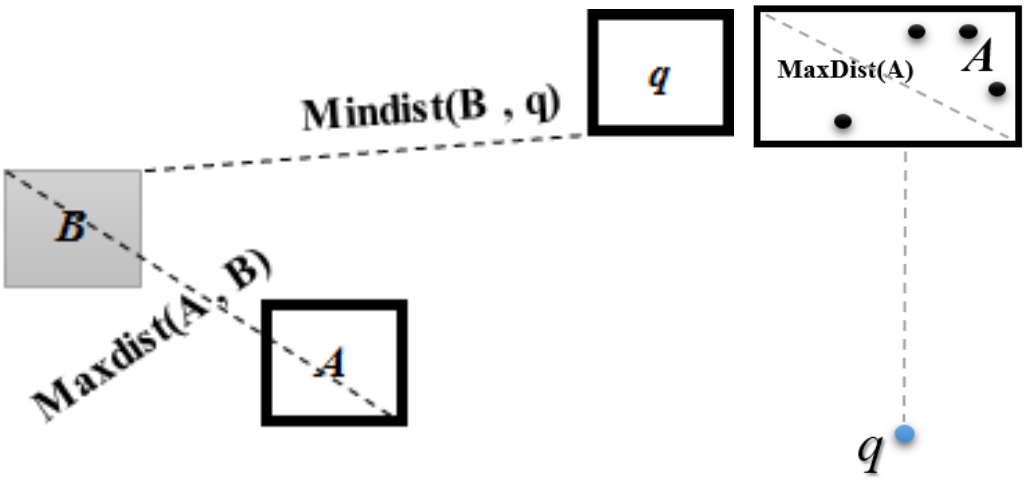
The key to the query processing algorithm is to minimize the number of PIR requests (block retrievals) and hence reduce the overall query cost. Our main idea is to use the summary information (DB_{cnt}) to prune cells or regions that do not contain query answers as much as possible, hence avoid unnecessary retrievals. Our query answering algorithm is presented in Algorithm 3 based on RkNN-HG indexing method. This algorithm can be also applied on RkNN-HRT with minor modifications discussed in Section 4.3.5. Our approach entails two phases, *pruning* (line 3-25 of the algorithm) and *verification* (line 27-31). The pruning phase restricts the search space of finding the answer while the verification phase utilizes the actual location of points in the restricted space to answer the query.

When a user issues a RkNN query, she locates the query cell Q based on the query location q and the grid/map specifications provided by the server (line 2). The rest of the algorithm can be summarized as follows. First, the user utilizes the information in DB_{cnt} to select the cells that may include her answer (line 3-25). She computes the indexes of the data inside her chosen cells in DB_{loc} and pads her request with dummy indexes if necessary, so she can retrieve cnt_{loc} blocks of location coordinates from DB_{loc} (line 26). After that, using the retrieved location coordinates of data points, she refines the set of reverse k-nearest neighbors for her query location q (line 27-31 of the algorithm). Finally, she retrieves the additional information about her answer set from DB_{dtl} using the pointers provided in DB_{loc}. Once again, she may need to pad her request with dummy indexes to be able to obtain cnt_{dtl} blocks from DB_{dtl} (line 32). In the rest of this section, we present the details of our approach to answer RkNN queries based on Algorithm 3.

4.3.1 Pruning Strategies

The most important part of our algorithm is the pruning phase which shrinks the search space into cells that might include the answer of the RkNN query (i.e. candidate cells), or might influence them (i.e. influence cells). To exploit the information provided in DB_{cnt}, we utilize three state-of-the-art uncertain pruning strategies which

are able to prune the search space using aggregated counts of data in each cell. These methods include i) *Min-Max* [10], ii) *60-degree* [66], and iii) *Half-plane* [17] which are applied in the order of their efficiency and the reverse order of their pruning power. We use such ordering because Min-Max is simple and more efficient, however it has less pruning ability compared to 60-degree method, so if Min-Max fails to prune a cell, we apply 60-degree method; similarly, half-plane may substitute 60-degree for the same reasons. In all of these methods, if an object B is pruned by another object A with regard to a query q, it means that B cannot be an answer for q as a result of A.



(a) Min-Max pruning

(b) Self pruning

Figure 4.5: Uncertain Min-Max pruning methods

Min-Max Pruning In Min-Max method [10], for a given query q, an object A can prune another object B if

$$MaxDist(A, B) < MinDist(B, q)$$

Figure 4.5a illustrates an example of Min-Max method. A special type of Min-Max pruning is called *Self-pruning* in which a cell is pruned by itself as shown in Figure 4.5b.

Self-pruning For an uncertain object A and query point q, if the number of points in A is greater than k, and

$$MaxDist(A) < MinDist(A, q)$$

then A is self-pruned [100]. MaxDist denotes the maximum distance between any possible points in A (e.g. diameter in a rectangle) and MinDist denotes the minimum distance between A and the query point q.

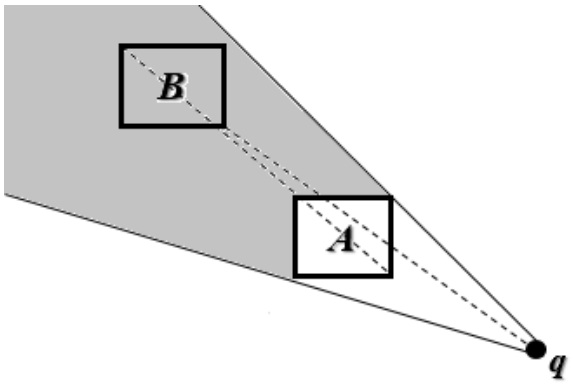
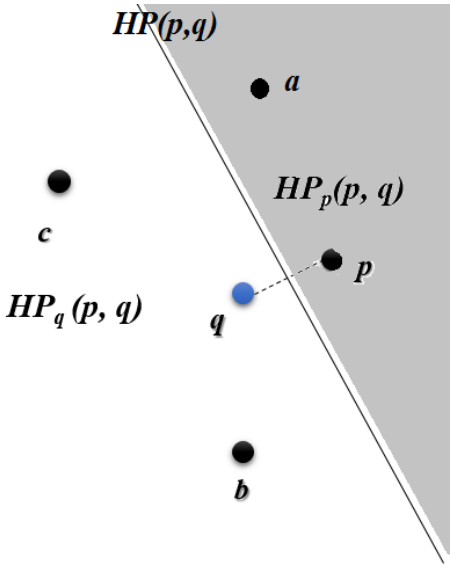


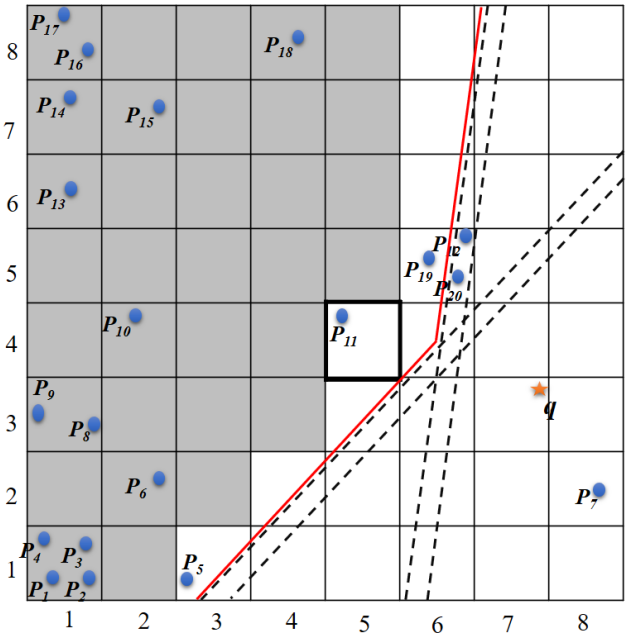
Figure 4.6: Uncertain 60-Degree pruning

60-Degree Pruning Li et.al [65] developed a 60-degree pruning technique for uncertain data. Given two uncertain objects A, and B, and a query point q, B can be pruned by A if i) for each point $a \in A$ and $b \in B$, $Dist(a, q) < Dist(b, q)$, ii) the angle range of A w.r.t q is less than 60°, and iii) the angle range of B w.r.t q is contained in the the angle range of A w.r.t q. An example of 60-degree method is shown in Figure 4.6.

Half-plane Pruning In exact half-plane method [93], the data space between a query q and data point p is divided into two planes by the perpendicular bisector $HP_{q,p}$; The half-plane $HP_p(q, p)$ contains p and $HP_q(q, p)$ contains q. This strategy is depicted in Figure 4.7a, any points falling into the half-plane in the side of p (i.e. $HP_p(q, p)$) is pruned with regard to q since they are closer to p than q.



(a) Exact method



(b) Uncertain method

Figure 4.7: Half-plane pruning methods

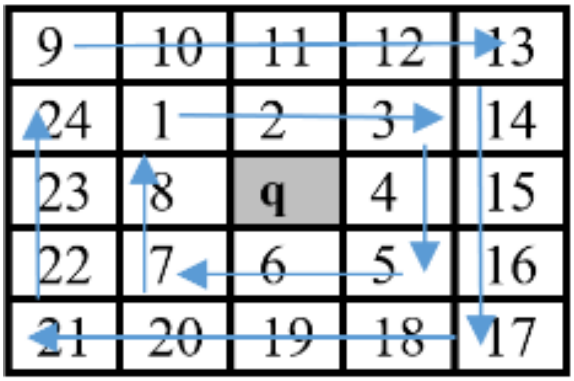


Figure 4.8: Order of access to cells around q

In a situation where the exact locations of data points are not known like grid cells in our method, half-plane method is extended to be able to prune uncertain regions [17]. Given an uncertain region A and a query point q, the boundary of pruned area can be computed as intersection of the four bisectors between q and each vertexes of region A.

Figure 4.7b illustrates the uncertain half-plane pruning method which divides the space into two half-planes for each vertex of cell $< 5, 4 >$. The intersection of four half-planes prunes all of the gray cells.

4.3.2 Pruning Algorithm

The cells that need to be retrieved from the server after pruning consists of 1) candidate cells which may include the RkNN answers, and 2) influence cells for each candidate cell in order to verify if the points in the candidate cells are indeed the RkNN answers. We describe in detail below how to compute the candidate cells and influence cells for each candidate cell.

Candidate cells are all non-empty cells that may include the answer to a RkNN query if the actual coordinate of points in them become available.

As illustrated in Figure 4.8, our pruning phase accesses the cells around the query in clockwise order and checks each cell against a set of *candidate cells* to see whether

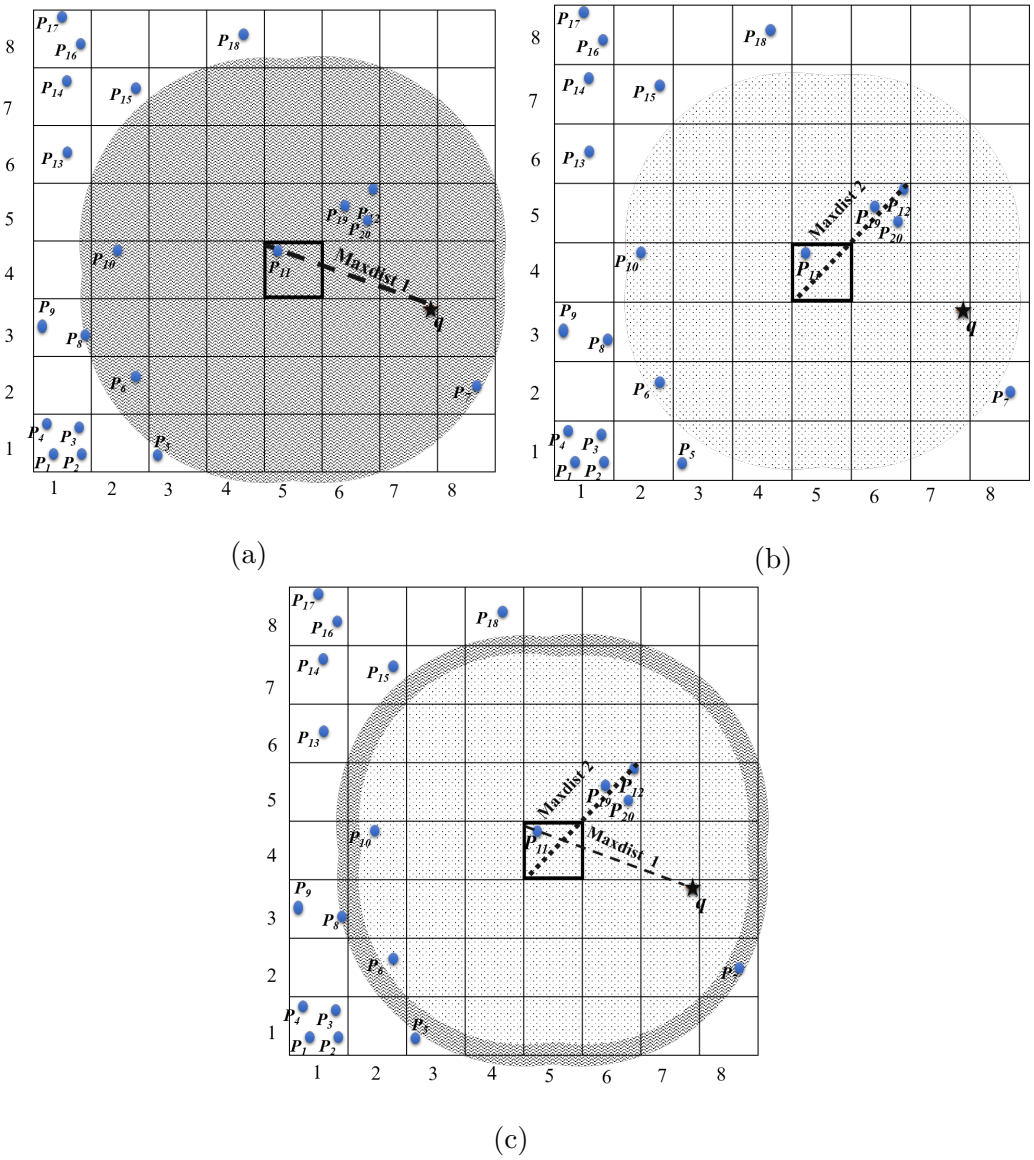


Figure 4.9: Computing influence cells for the candidate cell $< 5, 4 >$

they are pruned (line 7-25 of the Algorithm 3). At the beginning, the candidate set CND_c is empty or includes the query cell Q if it is non-empty. As pruning phase continues, any cell which is not pruned k times by candidate cells is added to the set. We keep track of the number of times a cell is pruned by incorporating the number of data points in pruning cells. It means, if a candidate cell *cnd* which has 3 points prunes c with regard to the query q, we record c is pruned 3 times ($c.pruned = 3$). If all of the cells in a layer around the query are pruned k times, the search stops, otherwise, pruning phase continues to assess upper layers. After retrieving all of candidate cells, the pruning phase is continued by computing *influence cells*.

Influence cells for every candidate cell is the set of all non-empty cells that do not prune candidate cells, however, if the actual coordinate of points in them are realized, they may prune the points in the candidate cell.

We compute a set of influence cells INF_c for every candidate cell *cnd* as follows. First, we probe the cells around *cnd* and retrieve every cell c that does not prune the candidate cell, but satisfies the following condition [65].

$$MaxDist(q, cnd) \geq MinDist(c, cnd) \tag{4.1}$$

We add c to a set called INF_1. This means, c may include points which are closer to q compared to *cnd*. Figure 4.9a illustrates the cells in INF_1 for the candidate cell $cnd =< 5, 4 >$ when $k = 1$. It can be easily computed as the area surrounding the cell $< 5, 4 >$ with distance $Maxdist_1$ from every possible point in the cell where $Maxdist_1 = MaxDist(q, cnd)$. While INF_1 assures the inclusion of all possible influence cells [65], it might be redundant in certain situations. To further refine the influence cells, we can remove any influence cell that does not belong to the possible set of kNN neighbors of the cell *cnd*. In other words, any cell c that doesn't satisfy

the following condition cannot be an influence cell for *cnd*.

$$\nexists R \subset Cells,$$
$$\max_{r \in R}(MaxDist(r, cnd)) < MinDist(c, cnd) \quad (4.2)$$
$$\text{and} \quad \sum_{r \in R} r.count \geq k$$

which means there are a subset of cells R which are all closer to the candidate cell *cnd* compared to c and they have at least k points combined. The intuition is that, the cells in R are among k nearest neighbors of the candidate cell *cnd*, so any cell that is absolutely farther than them from *cnd* cannot possibly be among kNN neighbors of *cnd* and therefore an influence cell of it. Figure 4.9b illustrates INF_2 as the shaded area which intersects all of the possible influence cells that satisfy the condition above for the candidate cell $< 5, 4 >$ when $k = 1$. It is computed as the area surrounding the cell $< 5, 4 >$ with distance $Maxdist_2$ from every possible point in the cell such that $Maxdist_2 = \max_{r \in R}(MaxDist(r, cnd))$ where R includes the closest cells to *cnd* that have k points collectively. Clearly, if q is not in INF_2 (i.e. among *cnd*'s possible kNN), then *cnd* cannot be a candidate in the first place.

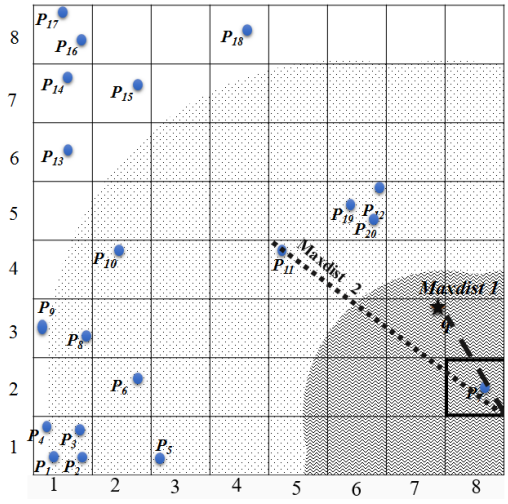


Figure 4.10: Computing influence cells for the candidate cell $< 8, 2 >$

Theorem 4.4. *Given a query point q, a candidate cell cnd, INF_1 as the set of cells satisfying Condition 4.1 and INF_2 as the set of cells satisfying Condition 4.2, the influence cells for cnd is computed as the set $INF = INF_1 \cap INF_2$.*

Proof. Figure 4.9c illustrates INF as the intersection of areas covering INF_1 and INF_2 for the candidate cell $cnd =< 5, 4 >$. Let's assume there is a cell c which is an influence cell for the cell *cnd*, but $c \notin INF$. Since c is an influence cell of *cnd*, it satisfies both Conditions 4.1 and 4.2, so $c \in INF_1$ and $c \in INF_2$. Assuming the case in Figure 4.9c which indicates $Maxdist_1 > Maxdist_2$, we can easily see that $INF_2 \subset INF_1$ because both INF_1 and INF_2 are areas centered at the cell *cnd* with the radius of $Maxdist_1$ and $Maxdist_2$. As a result, we have $INF = INF_1 \cap INF_2 = INF_2$ and since $c \in INF_2$, we will have $c \in INF$ which contradicts the assumption and concludes the proof. A similar conclusion can be easily drawn for the case when $Maxdist_1 < Maxdist_2$ as illustrated in Figure 4.10 for the candidate cell $< 8, 2 >$. Similarly, the proof applies when $Maxdist_1 = Maxdist_2$. □

4.3.3 Verification Phase

Once the pruning phase is complete, the location coordinates of candidate and influence cells are retrieved from DB_{loc}. After retrieving candidate and influence points, a verification check is applied on them to refine the candidate points and obtain the final query results.

Candidate Verification A candidate point *cnd* cannot be in the RkNN set of the query q, if the range query centered at *cnd* with radius of $Dist(cnd, q)$ contains greater than or equal to k points.

When final results are determined, actual data points along with their complete information are retrieved from DB_{dtl} through pointers stored in DB_{loc}.

4.3.4 RkNN Query Retrieval Example

Figure 4.3 illustrates the map and the indexing structure for the example data space. Client issues a RkNN query for $k = 1$ at the query point q shown as a star. Using the map and grid specifications, she learns that q is located at cell $< 7, 3 >$ with the Hilbert-curve value of 51. With the help of aggregated information stored locally in DB_{cnt}, she easily retrieves the information of query cell $< 7, 3 >$. The pair of information associated with this cell is $< 18, 0 >$ which indicates that there is no point in this cell and there exist 18 points in the preceding cells in the order of Hilbert curve.

Afterwards, client runs the pruning algorithm with the set of candidate cells empty at the beginning. She checks each cell around the query in clockwise order, so the first non-empty cell that she reaches is $< 8, 2 >$ and runs the pruning methods (since the set of candidate cells is empty, the only applicable method is self-pruning method). Since $< 8, 2 >$ is not pruned by any candidate cell or by itself, it is added to the set of candidate cells. The search continues for other cells while they are checked against the candidate cells. When the pruning phase is finished, the set of candidate cells includes $< 8, 2 >$ and $< 5, 4 >$. Moreover, the influence cells for each of these candidate cells are $\{\}$ for $< 8, 2 >$ as shown in Figure 4.10, $\{< 6, 5 >, < 8, 2 >, < 3, 1 >, < 2, 2 >, < 2, 4 >\}$ for $< 5, 4 >$ as shown in Figure 4.9c.

In the verification phase, the client retrieves the point coordinates for candidate set and their influence sets from DB_{loc} by asking for blocks $\{B_{2,2}, B_{2,4}, B_{2,5}\}$ through PIR. Block indexes are computed using the count information provided in DB_{cnt}. As a result, location coordinates of points $\{P_7, P_{11}\}$ are retrieved as candidate points with influence points as $\{P_5, P_6, P_7, P_{10}, P_{12}, P_{18}, P_{19}\}$. In the verification phase, client observes that P_{19} is closer to P_{11} than q, so, P_{11} cannot be an answer. Thus, $R1NN(q) = P_7$.

Finally, the client retrieves the detailed information of P_7 from DB_{dtl} through the pointer stored in DB_{loc}. Therefore, she asks for block $B_{3,7}$ through PIR.

4.3.5 Query Answer Retrieval Modifications for RkNN-HRT

Query answering in RkNN-HRT is similar to the described methods for RkNN-HG with some differences. In this section, we highlight the necessary modifications to Algorithm 3 to work for Hilbert R-Tree indexing structure using the example in Figure 4.4. First, Q is located as cell c_{11} using the *locateQueryCell* method similar to RkNN-HG.

Instead of examining grid cells, we only check leaf nodes in this method, so the candidate and influence cells are substituted by leaf node MBRs. The pruning phase also uses a different search strategy while pruning methods are the same. Since leaf node summaries in DB_{cnt} include both minimum bounding rectangle mbr and the largest Hilbert values of their data lhv, we can search for candidates by examining the closest leaf nodes based on minimum Euclidean distance or Hilbert values. In our example, if we search using Hilbert values, the algorithm finds R_1 as the closest leaf since its Hilbert value $R_1.lhv$ is 3; Alternatively, if Euclidean distance is used, $R_2.mbr$ has the minimum distance to q and is examined first. Our experiments showed no significant difference between these two strategies, so we only used Hilbert values for more efficiency. The stopping criteria for the pruning phase is also changed since we do not examine empty spaces any more, so all of the leaf nodes are probed before the algorithm stops. However, by examining the closer nodes first which are more likely to be candidates, many of the farther nodes will be pruned by them reducing the cost for computation of half-planes. We use the same pruning methods as RkNN-HG. The rest of the algorithm remains the same.

4.4 Query Plan Pre-computation

In order to preserve the privacy of RkNN queries and ensure that client receives accurate answers, we propose an offline algorithm which computes the query plan to be used by all clients regardless of their location. This algorithm finds the maximum number of PIR retrievals performed in DB_{loc} and DB_{dtl} for any RkNN query q and

stores it in a tuple as $< cnt_{loc}, cnt_{dtl} >$.

For computing the query plan, the server uses fine-granined grid cells and treats them as an uncertain query, then using a modified version of query answer retrieval method described in Section 4.3, it retrieves the RkNN set for the uncertain query. Using this approach, the retrieved number of data blocks from DB_{loc} and DB_{dtl} for an uncertain query Q_u is guaranteed to be sufficient to answer RkNN queries issued for any query point q where $q \in Q_u$. In this section, we describe the required modification to query answer retrieval algorithm.

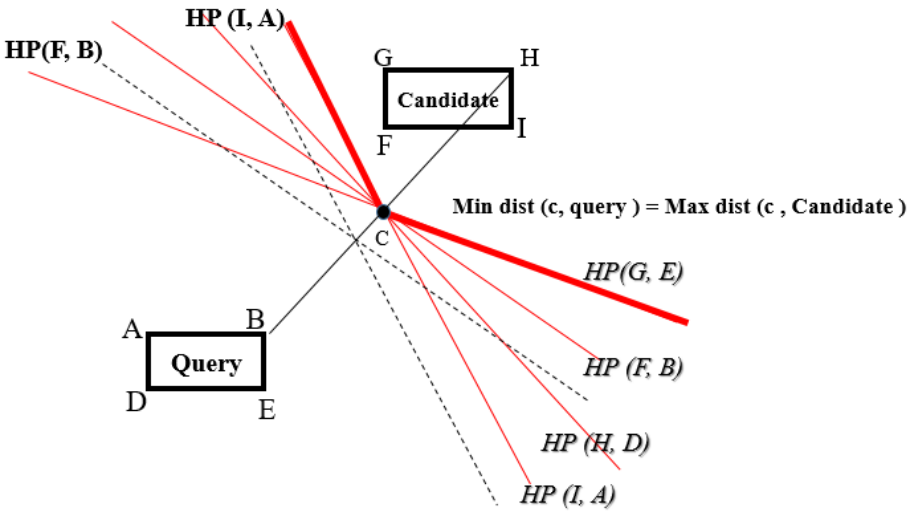


Figure 4.11: Uncertain half-plane pruning

4.4.1 Modified Pruning for Query Plan Computation

To be able to prune uncertain rectangular areas (i.e. grid cells or MBRs) against an uncertain query, we cannot utilize the uncertain pruning methods described in Section 4.3, therefore, we adopt a solution based on half-plane pruning method proposed in [18]. Figure 4.11 shows the computation of pruned area when both query and points are uncertain. Two uncertain rectangles represent the uncertain query Q_u and an uncertain candidate *cnd*. In contrast to the uncertain half-plane pruning method

described in Section 4.3, we cannot define the boundary of pruned area by merely intersecting the four bisectors, because we may miss some points as answers [18]. To compute the pruned area, we find a point o in which $Mindist(o, Q_u) \geq Dist(o, cnd)$, and pass all of the bisectors between the query and the candidate rectangles through this point; The intersection area of all these bisectors is defined as the pruned area.

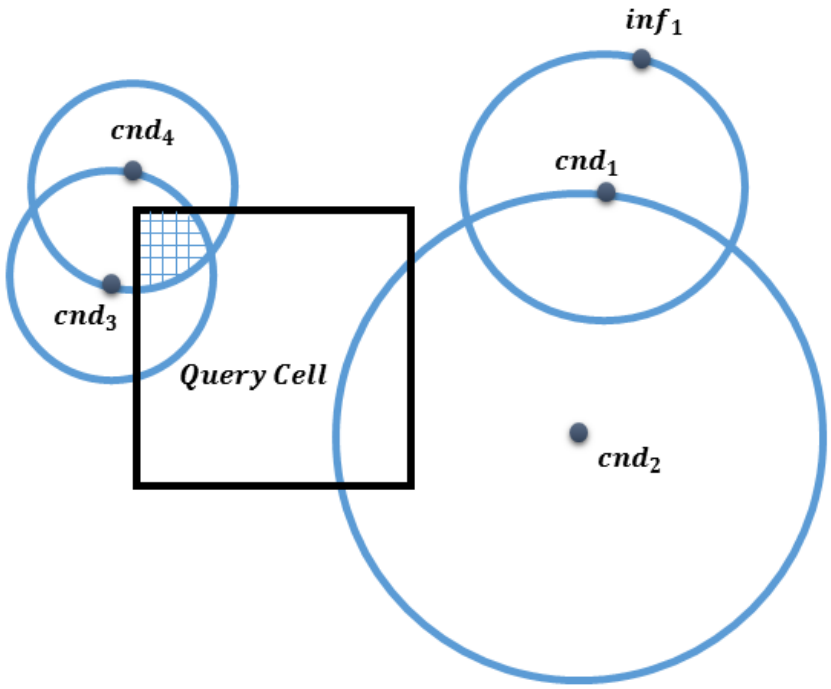


Figure 4.12: Modified verification for query plan

4.4.2 Modified Verification for Query Plan Computation

After retrieving the location coordinates of all candidate and influence cells from DB_{loc}, we need to refine the candidate points to obtain the final query answers. Since the actual location of query is not known, we cannot verify the candidate points similar to Section 4.3, so instead we propose a method to find all plausible answer sets.

First, we compute the distance between each candidate point and their k nearest neighbor among the retrieved points and we call it $kthDist$. If a circle centered in the candidate point cnd with the radius of its $kthDist$ intersects the query cell, cnd is a possible answer for points in the intersected portion of the query cell. By drawing similar circles for all candidate cells, we can find possible answer sets for different regions of query cell. Finally, we retrieve the possible answer sets from DB_{dtl} and record the maximum number of retrieved blocks among them. Figure 4.12 illustrates an example in which there are several possible answer sets $\{cnd_2\}, \{cnd_3\}, \{cnd_4\}, \{cnd_3, cnd_4\}$. Among the answer sets, $\{cnd_3, cnd_4\}$ collectively intersect with the query cell in the patterned area and has the highest number of page retrievals from DB_{dtl}, so is recorded.

4.5 Experimental Evaluation

We conduct several experiments with two real-world datasets to assess the effectiveness of our solutions based on the computational and overhead cost.

4.5.1 Experimental Methodology

We adopt rigorous models for simulating private DB block retrieval with secure hardware PIR (SCOP) which are based on [103], and [77]. To calculate the computational cost for retrieving a block of data through PIR we set the following parameters as described in [77]. Block-size is set to $4KB$, disk-seek as 5ms, disk_read/write as 100 MB/s, SCOP_read/write as 80 MB/s, and SCOP_encrypt/decrypt as 10 MB/s. We use the following metrics to evaluate our methods.

The computational cost (query answering time) for each query is calculated using the simulated PIR retrieval time for required blocks of data. The number of required blocks is specified by the pre-computed query plan (i.e. the maximum number of data blocks from each database necessary to answer any query).

The client-side storage overhead cost is computed as the relative size of DB_{cnt}

Table 4.1: Summary of datasets

Dataset	Gowalla	California
Size of DB_{loc}	5MB	26MB
block retrieval time (s) in DB_{loc}	0.47	0.71
Size of DB_{dtl}	20MB	104MB
block retrieval time (s) in DB_{dtl}	0.66	0.94

with regard to the size of DB_{dtl} or in other words $\frac{|DB_{cnt}|}{|DB_{dtl}|}$.

As described in the previous sections, query plan is pre-computed by the server, hence we do not report its computation time.

4.5.2 Datasets

We tested our methods using two real datasets Gowalla [1] and California [64]. Gowalla contains check-in information of users of a location-based social network in New York. The check-ins consist of time and location coordinates of users at different points of interests (POIs). We use the coordinates of 19483 points in our experiments. California dataset contains the location coordinates of 104770 points of interest in California. Table 4.1 shows the summary of databases created for each dataset along with the computational cost for retrieval of a block of data from each database (i.e. per PIR Request). The size of data points (i.e. description, images, etc) is set to $1KB$ in all of our experiments.

4.5.3 Parameter Tuning

Table 4.2 shows the parameter settings with default values highlighted with an underline for Gowalla dataset and in bold font for California dataset. We use the grid granularity $g_{QP} = 512$ to compute the query plan in all of the methods. We tune the grid granularity for the indexing and query processing and other parameters for each method as follows.

Table 4.2: Parameter settings

Parameter	Values
k	**1** 2 3 4 5
g_n (Naive Grid Granularity)	**32** 64 128 256
g (RkNN-HG Grid Granularity)	32 64 128 256 **512**
h (RkNN-HRT Hilbert Resolution)	32 64 128 256 512 **1024**
b (Node Block Size (B))	32 64 128 **256** 512
t (MBR Size Threshold)	32 64 128 **256** 512
g_{QP} (Query Plan Grid Granularity)	**512**

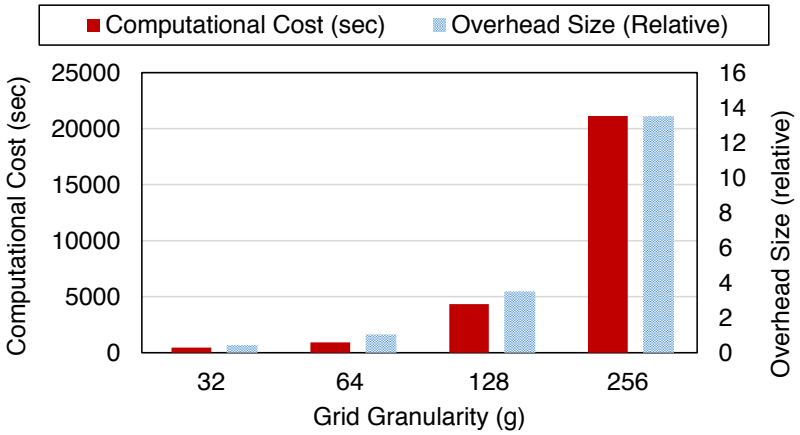
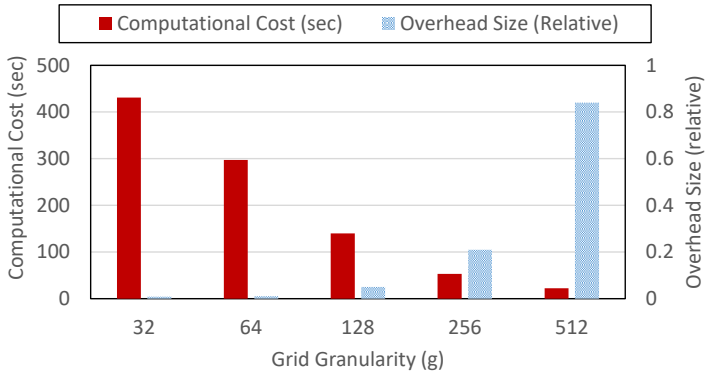


Figure 4.13: Naive: The effect of grid granularity g_n (Gowalla)

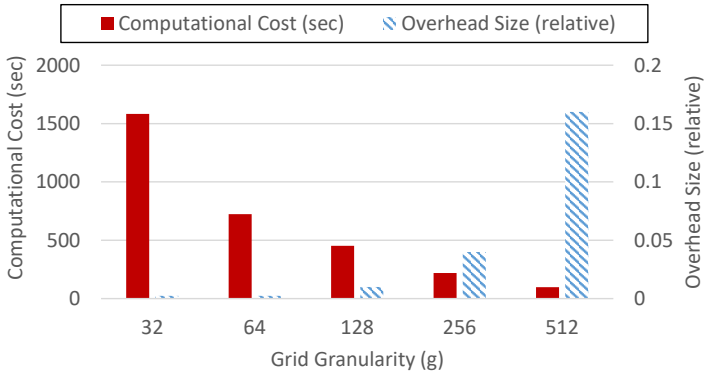
Naive Approach

Figure 4.13 shows the effect of the parameter g_n (grid granularity) on query processing and answer retrieval time for the Naive approach using the Gowalla dataset. Looking at the computational cost, with minimal cost of 452 seconds, it is clear that this method is not efficient. We have also demonstrated the size of DB_{loc} relative to the size of DB_{dtl} for every values of g_n to show how the size of this database is affected by grid granularity in this method. As g is increased, the query answering time is increased. The size of DB_{loc} is also increased reaching up to 14 times the size

of original data for $g_n = 256$. Due to similar highly inefficient results for California dataset, the figures are excluded.



(a) Gowalla dataset



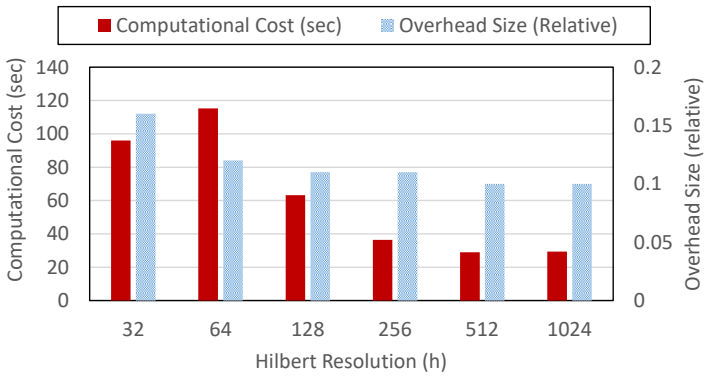
(b) California dataset

Figure 4.14: RkNN-HG: The effect of grid granularity g

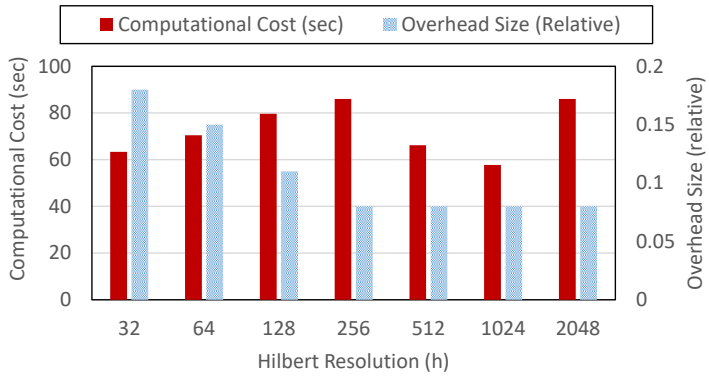
RkNN-HG

Figure 4.14 shows the effect of the parameter g (Grid granularity) on query processing time for RkNN-HG using both Gowalla and California datasets. In addition to processing time, we have also included the relative size of DB_{cnt} with regard to DB_{dtl}. Note, the size of DB_{loc} is determined by the size of dataset and thus is independent of the settings (see Table 4.1). Using Hilbert curve for storing the

points in the cells is the reason for smaller DB_{loc} in this method compared to Naive method. As g increases, the processing time decreases while the size of DB_{cnt} is increasing. Based on Figure 4.14a, we choose $g = 256$ as the best granularity for Gowalla dataset with an acceptable DB_{cnt} value. Figure 4.14b shows $g = 512$ is the best setting for California data.



(a) Gowalla dataset



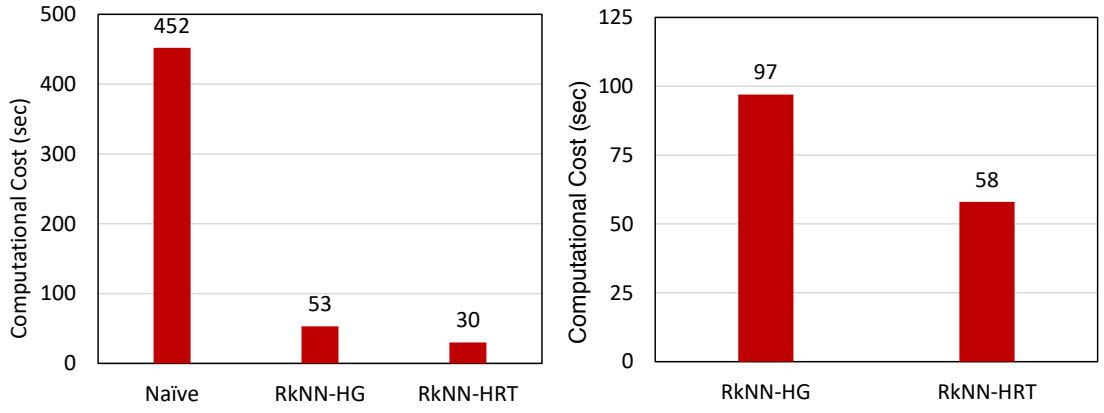
(b) California dataset

Figure 4.15: RkNN-HRT: The effect of Hilbert resolution h

RkNN-HRT

We set the node block size b as 128 bytes for Gowalla dataset and 256 bytes for California dataset. The node block size limits the number of children per node in

the HR-Tree. Additionally, we modified the HR-Tree algorithm to use a threshold $\frac{1}{t}$ which limits the size of the MBRs as a fraction of the map size. This parameter is set as 64 for the Gowalla data which means the diagonal of MBRs are less than $\frac{1}{64}$ of the map diagonal. Parameter t is set as 256 for the California data respectively. Figure 4.15 shows the effect of Hilbert resolution (h) on computational cost and the relative size of DB_{cnt} with regard to the size of DB_{dtl} (i.e. the original data). As shown in the figures, increasing h does not necessarily reduce the computational cost, but decreases the size of DB_{cnt}. We choose $h = 512$ for Gowalla and $h = 1024$ for California as the best settings resulting in the smallest computational and overhead cost.



(a) Gowalla dataset

(b) California dataset

Figure 4.16: Comparison of methods

4.5.4 Evaluation

Comparison of Methods

After tuning each method with their best parameter settings, we compare them side by side in Figure 4.16. In Figure 4.16a, we observe that RkNN-HRT and RkNN-HG both significantly outperform the naive methods, and HRT further outperforms HG

with computational cost of 30 seconds per query ($k = 1$) based on the computed query plan (i.e. maximal query processing time). Similarly, Figure 4.16b compares the computational cost of RkNN-HRT and RkNN-HG for California dataset omitting the Naive approach. Again, RkNN-HRT performs better than RkNN-HG.

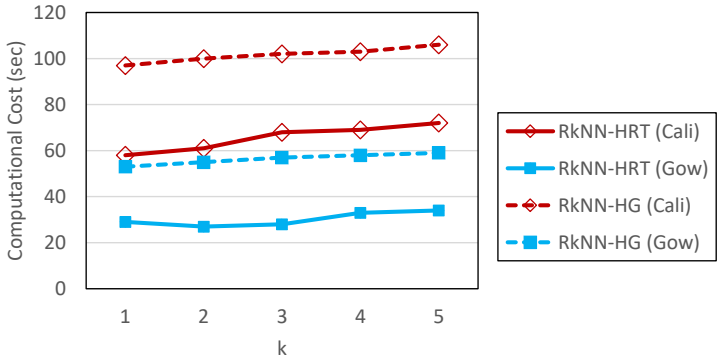


Figure 4.17: The effect of parameter k

The Effect of Parameter k

Figure 4.17 illustrates the effect of parameter k on RkNN-HRT and RkNN-HG. As we see, for both datasets, our methods scale well with k.

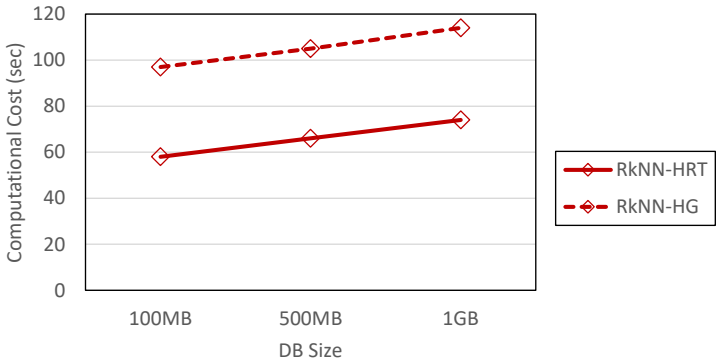


Figure 4.18: The effect of DB size

Scalability Analysis

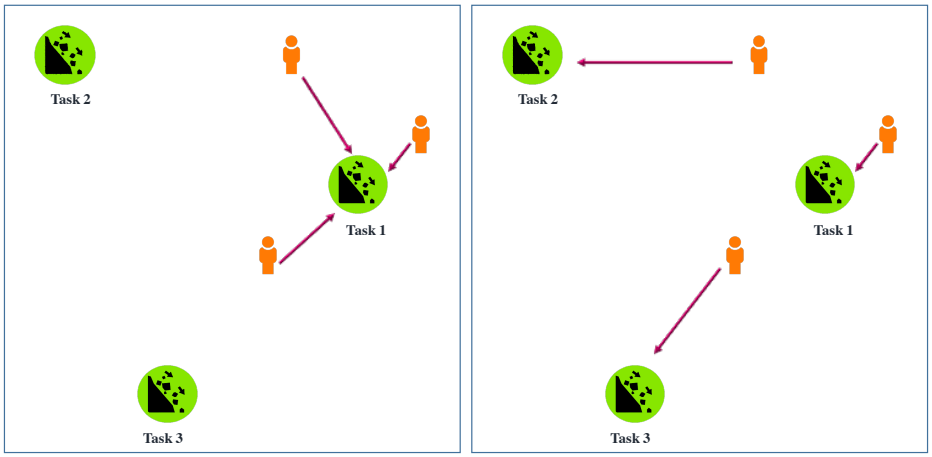
To study the effect of database size on processing time, we increased the data point size from $1KB$ up to $10KB$ for California dataset expanding the size of DB_{dtl} from approximately $100MB$ to $1GB$. Note DB_{cnt} and DB_{loc} are not affected by the size of data points. Figure 4.18 presents the results of varying the size of data on computational cost. While the size of data is increased by a factor of 5 and 10, computational cost is not affected by the same scale.

4.6 Application of Private RkNN Queries for Autonomous Spatial Task Selection

RkNN queries have many applications in various domains such as location-based services, marketing, and outlier analysis [57, 89, 93, 94]. In particular, this query can be utilized by volunteers of a geo-spatial crowd sensing application for task selection. Figure 4.19 shows an example of volunteers after a disaster who are willing to report the condition of affected areas for collective mapping of damages. In an autonomous task selection scheme, each volunteer may choose its nearest tasks as shown in Figure 4.19a which results in unbalanced reporting of damages (i.e., no one reports tasks 2 and 3 while three volunteers report task 1). Alternatively, they can choose their reverse nearest tasks as shown in Figure 4.19b for more balanced coverage.

In this section, we evaluate two methods for autonomous spatial task selection including NN and RNN queries. We show that RNN outperforms NN resulting in higher task coverage and task cost. **Total Task coverage** includes the total number of distinct tasks completed by participants. **Average Task Cost** is the average distance traveled by participants per completed task.

Spatial Task Selection Using NN Queries In this method, each participant selects her nearest task from the pool of available tasks in the server and performs



(a) (b)

Figure 4.19: Using (a) NN and (b) RNN for autonomous task selection

it. We assume each participant is able to perform only one task.

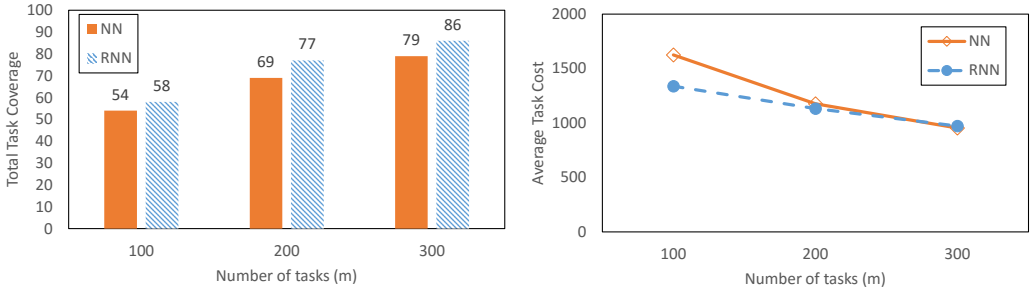
Spatial Task Selection Using RNN Queries In this method, each participant selects the closest task from the set of its RNN tasks retrieved from the pool of available tasks in the server and performs it. Again, we assume that each participant is able to perform only one task.

4.6.1 Evaluation Setting

We generated a synthetic data set including 1000 tasks and 1000 participants uniformly distributed in a map. In each experiment, we select n participants and m tasks randomly from the data set and evaluate both task selection methods. We compute the total task coverage and average task cost in each experiment.

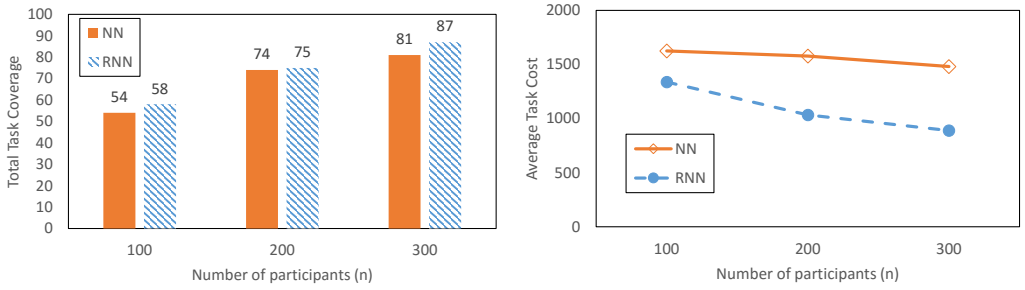
4.6.2 Results

The Impact of the Number of Tasks Figure 4.20 shows the result of our experiment when we vary the number of tasks for a fixed number of participants (i.e.



(a) (b)

Figure 4.20: Impact of the number of tasks on (a) total task coverage, and (b) average task cost



(a) (b)

Figure 4.21: Impact of the number of participants on (a) total task coverage, and (b) average task cost

100 participants). In Figure 4.20a, we observe that RNN yields higher total task coverage than NN for varying number of tasks. While NN guarantees that every participant always retrieve a task, the lack of a coordinator results in several participants selecting the same task which leads to over-coverage. Over-coverage causes high average cost per completed task for NN as shown in Figure 4.20b. However, as we increase the number of tasks, the chance of over-coverage decreases for NN which results in lower cost. RNN does not guarantee a task per participant, however, it does not suffer from the over-coverage problem either.

The Impact of the Number of Participants We also vary the number of participants for a fixed number of tasks (i.e. 100 tasks). Figure 4.21 shows that RNN outperforms NN for both task cost and coverage. As we increase the number of participants, the chance of over-coverage increases for NN resulting in a higher cost.

Chapter 5

Dynamic Data Driven Crowd Sensing Task Assignment

In this chapter, we propose a dynamic data driven framework for spatial task assignment in mobile crowd sensing with dynamic and uncertain participant locations. Our approach is based on the DDDAS (Dynamic Data Driven Application Systems) [25] paradigm. The DDDAS concept is crucial to address the big data problem in such crowd sensing applications in order to steer and assign the data collection tasks in targeted ways, adapting dynamically to application needs and the dynamic and uncertain locations of participants. The task assignment entails a synergistic feedback loop between application simulations and data collection: 1) based on assigned tasks, participants report the collected data and possibly their current (uncertain) locations to the application; 2) the collected data are dynamically integrated into an executing simulation to augment or complement the application model (e.g. flood movement), 3) the reported (uncertain) locations are dynamically integrated into an executing mobility model to accurately track participants' moving locations, and 4) conversely the executing simulations update the data collection targets and requirements as well as participants' locations which are then used by the task assignment module to make new task assignments and steer future data collection. Through model-based prediction and filtering, the DDDAS feedback loop is essential to dynamically steer future data collection, adapting in real time to data dynamics, moving participants, and application needs.

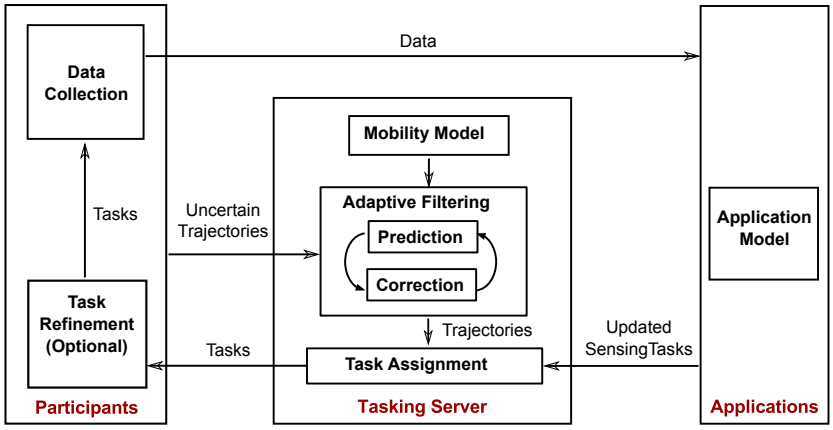


Figure 5.1: Adaptive dynamic data driven framework for uncertain spatial task assignment

5.1 Dynamic Data Driven Framework For Task Assignment

We propose a novel DDDAS based framework for dynamic spatial task assignment with uncertain trajectories illustrated in Figure 5.1. Our framework includes an offline learning process which builds a mobility model by mining the public trajectories to be used as the process model in the filtering module. The framework entails a feedback loop composed of the following key components:

- Participants, based on their assigned tasks, report collected data and voluntarily their current (uncertain) locations to the application.
- The collected data are dynamically integrated into an executing simulation to augment or complement the application model (e.g. flood movement), which updates the sensing targets for future data collection,
- The reported (uncertain) locations are dynamically integrated into a filtering component to augment or complement a mobility model. The prediction step

computes or simulates a participant's current location based on her historic trajectory and the mobility model (i.e. prior estimates). The correction step integrates the reported (uncertain) location and predicted location into a more accurate location (i.e. posterior estimate).

- The updated sensing targets and requirements as well as participant information with posterior estimates of their locations are then fed to the uncertain task assignment module which assigns tasks to participants using a probabilistic model while globally optimizing sensing coverage and cost. The output of this module, which is a mapping of tasks to participants, will be returned back to participants.

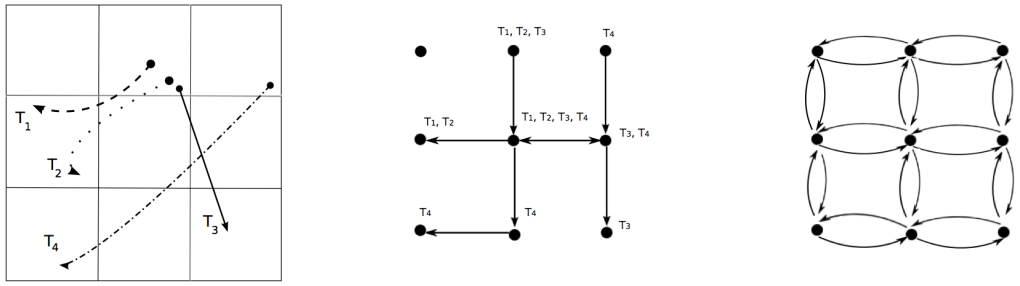
- Once each participant receives a set of tasks from the tasking server, if she has access to her exact location, a local task refinement step (i.e. a second-stage tasking module) can be used to further optimize her set of tasks.

- The final assigned tasks are then used to steer the future data collection as well as possibly participants' future trajectories since they might need to travel to the location of sensing targets.

Since the application model is dependent on specific applications and sensing tasks, we focus on the general task assignment module in this chapter and explain each component in detail below.

5.1.1 Learning Mobility Models

To learn a mobility model, we use a set of publicly available trajectories as a historical data set to calculate transition probabilities between adjacent locations. To formulate the location transition process, Bayesian inference and Markov model are two popular methods which are examined in this chapter. Note that, we do not consider modes of transport or location-based activities to build these models, however this module can be easily extended to include them. We note that we can learn mobility models



(a) A 3×3 grid representation (b) A graph representation (c) A Markov model

Figure 5.2: An example map with a grid, graph, and Markov model representation

for specific individuals if we have sufficient historical personalized trajectories. Of course, additional privacy mechanisms may be required to protect the individual's privacy when learning such personalized models.

Bayesian Inference

A grid-based road network with mapped trajectories can be used to build a Bayesian inference framework for next location prediction [106]. Generally, a map is a two-dimensional $g \times g$ grid with granularity of a cell (i.e. all the locations within a single cell are considered to be the same). A graph is build based on the grid where each cell corresponds to a node in the graph, then trajectories are mapped to sequences of these nodes. An example of a 3×3 grid is given in Figure 5.2a, where trajectories T_1, T_2, T_3, and T_4 are shown in the map. Figure 5.2b represents the trajectories mapped in a graph which is created based on the grid. T_1 and T_2 are identical in the graph because of the granularity which is a grid cell.

After mapping the set of all trajectories T, given a partial trajectory T_j, the probability of a node n being the next destination of T_j can be computed based on the Bayes rule as Equation 5.1, which is the probability that node n contains the next location in given trajectory T_j, conditioning on the query trajectory T_j.

$$P(next(T_j) = n \mid T_j) = \frac{P(T_j \mid next(T_j) = n)P(next(T_j) = n)}{P(T_j)} \tag{5.1}$$

On the other hand, the probability that node n is the next destination of trajectory T_j can be calculated as Equation 5.2 which is the number of trajectories containing the sequence of nodes in T_j followed by node n divided to the total number of trajectories.

$$P(next(T_j) = n) = \frac{|\{T_i \mid (T_j, n) \in T_i\}|}{|T|} \tag{5.2}$$

Markov Model

Using the same graph as Figure 5.2b, a Markov model can be constructed which assumes a state for each node of the graph as in Figure 5.2c. For each pair of adjacent nodes, both transition directions are considered and the probability of each directed edge p_{rs} and p_{sr} are calculated as the probability of location transition between n_r and n_s and vice versa. In a first-order Markov model, only the current state determines the probability of transiting to the next state, so the probability p_{rs} is calculated as the number of trajectories which have the sequence of two nodes (n_r, n_s) divided to the number of trajectories which have n_r as shown in Equation 5.3.

$$p_{rs} = \frac{|\{T_i \mid (n_r, n_s) \in T_i\}|}{|\{T_i \mid n_r \in T_i\}|} \tag{5.3}$$

After calculating all the probabilities between the nodes, a $g^2 \times g^2$ transition matrix M is created which can be used as a process model in a state-space model as described in Section 5.1.2. Moreover, other advanced Markov models such as higher order or hierarchical Markov models can be built to feed more variables to the model which are not considered in this chapter, but can be easily plugged into the framework.

5.1.2 Adaptive Filtering

The filtering component in our framework provides estimates of noisy locations in order to improve the accuracy of location information per time stamp to be used in task assignment module. First, given the mobility model as described in Section 5.1.1, we can create a linear state space model as shown in Figure 5.3 and formulated in

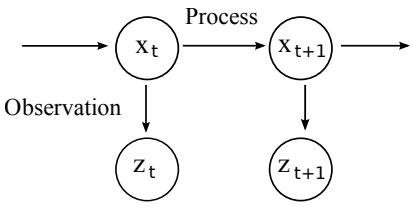


Figure 5.3: A Simple State-Space Model

Equation 5.4. In Figure 5.3, x represents true states while z shows observed states.

$$x_{t+1} = Ax_t + \omega_t \tag{5.4}$$
$$\omega_t \sim \mathcal{N}(0, Q)$$

where A is the time-invariant, linear coefficient and ω_k represents the noise of the linear model. Intuitively, the process model linearly relates the current location state x_{t+1} to the previous state x_t as described in Markov model, except for a white Gaussian noise ω, called the process noise with variance Q. The observed state z_t is also obtained from the true state x_t at each time point t and contains additive measurement noise. we can build a observation model as Equation 5.5.

$$z_t = Hx_t + \nu_t \tag{5.5}$$
$$\nu_t \sim \mathcal{N}(0, R)$$

where H is the linear coefficient and ν_k represents the additive measurement noise. In our context, this noise can originate from device inaccuracy (e.g. GPS noise) or a perturbation method (e.g. differential privacy) and be modeled differently according to the measurement equipment or process. In this chapter, we assume a Gaussian noise with variance R.

Given the process model and the measurement model, a filtering algorithm is used for posterior estimation of true state to minimize the measurement error. Two popular filtering algorithms in literature are Kalman Filter [45] and Particle Filter [39]. Kalman filter is optimal for linear processes with a Gaussian noise, while Particle

filter makes no assumption about the process model or noise in the state-space model but can be computationally expensive.

5.1.3 Uncertain Spatial Task Assignment

The output of the filtering module for the given set of noisy participant locations at time t is a set of filtered uncertain locations to be used in spatial task assignment. Given a set of updated spatial tasks and assignment goals by applications, and the set of participants with uncertain locations (i.e. output of filtering module), we have developed methods and algorithms to handle location uncertainty and optimize assignment process to achieve required goals of the applications. In our work, we considered coverage-based assignment goals with a distance-based cost model, however, any assignment goal can be adopted in our task assignment module in the dynamic framework. General definitions of task assignment, spatial task and distance-based cost model is given as follows.

A *task assignment* is a mapping of participants to tasks. Each participant-task pair can be considered also an individual assignment. A cost might be set for each assignment.

A *spatial task* (i.e. location-based task) includes a target (i.e. an object, event, or phenomena), a location, a time-frame for sensing, and other specific instructions or sensor requirements to perform sensing.

A *distance-based cost model* defines the cost of each assignment as a function of distance between the participant and the task location. In the simplest definition which considers the exact distance as cost, closest participants to each task will form least costly assignments.

Task coverage could be defined in different ways.

- *Single-coverage* model means each task needs to be performed by only one participant to be considered as covered.
- *K-coverage* model requires each task to be assigned to k participants to be considered as covered. This model can be used in untrustworthy or uncertain environments to avoid faulty or missing data.

Considering these task coverage models, coverage-based assignment goals might fall in one of the following categories.

- *Maximum coverage assignment* aims at maximizing the coverage of tasks by participants. In a single-coverage model, the assignment goal can be summarized as maximizing the number of assigned tasks.
- *Minimum-cost coverage-based assignment* aims at achieving a task coverage goal with minimum cost. In a single-coverage model, the assignment goal might require only a portion of tasks to be assigned with minimum cost.
- *Minimum-cost maximum-coverage assignment* aims at maximizing a task coverage goal with minimum cost.

Task Assignment with Uncertain Locations In spatial task assignment with uncertain locations, since exact locations of participants are not provided, the distance information between task locations and participants is unavailable to the tasking server, therefore the server is required to deal with location uncertainty to achieve assignment goals. Spatio-temporal queries over uncertain data have been extensively studied with many algorithms to handle queries such as nearest neighbors, top-k, and range queries [102]. These queries mostly consider the results for one query object and not a set of objects, therefore cannot be directly adopted in our work or would be very inefficient. Among other spatio-temporal queries, K closest pairs query (K-CPQ) [23] which is the problem of finding the K closest pairs between two spatial datasets is the most related query type to our problem, but is not studied for uncertain datasets.

Knowing the set of uncertain locations of participants as a set of minimum bounding area of each uncertain location with a probability density function, we can apply two simple methods to calculate the expected distances between this set and the set of exact locations of the tasks. As a naive method, for each participant, we calculate the centroid point of all possible location samples in each uncertainty area and use it to calculate the expected distances between this point and all locations of the task set. As a more accurate alternative method, we first apply a geometric pruning algorithm to remove the task-participant pairs with zero probability of being accessible (i.e. a participant can not travel to the place of a task) and shrink the uncertain areas to only contain the accessible location samples. Then, for the remaining pairs with shrinked areas, we calculate the probabilities of the task locations being accessible by participants as well as the expected distance between task locations and shrinked areas. Finally, the set of expected distances and the accessibility probabilities can be used in our probabilistic task assignment methods proposed in Chapter 3.

Local Task Refinement Once the task assignment is done at the tasking server, the assignments are sent to individual participants. The goal of the local task refinement is to further optimize task assignment results of the global assignments by each participant using her exact location. This approach requires the participants to know their actual locations. This assumption is reasonable when spatial noise is added by participants for privacy-preserving purposes, but can not be applied to other types of uncertainty. Hence, we have considered an optional task refinement module in participant side of our proposed framework.

The final assigned tasks are then used by the participants to steer their future data collection, which completes the feedback loop. Our main insights are that given this feedback loop, data will be collected in a targeted way, adapting dynamically to application needs and data dynamics. The data collection process is adapted according to participants' moving trajectories and the sensing tasks in such a way that sensing cost and resource utilization are optimized.

Chapter 6

An Extensive Experimental Evaluation of Location Prediction Models For Moving participants

This chapter investigates and evaluates the existing state-of-the-art location prediction methods in an attempt to find suitable methods for modeling participants movements in our dynamic task management framework proposed in Chapter 5. Existing location prediction approaches target different applications with their parameters tuned for certain datasets. Many of such approaches have not been studied in variety of experimental settings with different data, thus, they lack an extensive evaluation to show their strengths and limitations in different situations. Moreover, due to existing of variety of spatial and temporal modeling of trajectories, several approaches are not comparable directly [73], therefore, we build a framework to evaluate and compare such methods extensively using different datasets and test parameters.

6.1 Problem Definition

We define the problem of *next location prediction* in this section as follows.

Definition 6.1. *A **trajectory** T is a sequenced set of tuples*

$$T = \langle \langle x_1, y_1 \rangle, t_1 \rangle, ..., \langle \langle x_n, y_n \rangle, t_n \rangle$$

where $\langle x_i, y_i \rangle$ *indicates a location at time point* t_i *and* $t_i \leq t_{i+1}$.

We use O to denote the set of trajectories for all moving objects while $O^{(i)}$ indicates the trajectories of one object.

Definition 6.2. *Given a set of trajectories* O*, and a partial trajectory of a moving object* P *at current time point* t_c*,*

$$P = \langle \langle x_1, y_1 \rangle, t_1 \rangle, ..., \langle \langle x_c, y_c \rangle, t_c \rangle$$

Next Location Prediction *problem predicts the object's location at next time point* t_{c+1}.

We assume all the trajectories are synchronized, otherwise, we interpolate the missing locations. We also assume equivalent time intervals between two consecutive time points in all trajectories Δt

$$\Delta t = t_{i+1} - t_i$$

which may vary for different applications. This assumption is necessary for our evaluations since several location prediction algorithms in this study do not handle variable length time intervals in their problems.

Other variants of problem formulation Several works formulate the location prediction problem as *path prediction with unknown destinations* [114]. Since the destination is unknown, the only difference between these definitions and our formulation is the number of subsequent locations that are predicted. Therefore, we include these methods in our evaluations.

There are several other location prediction approaches in literature which are not comparable to the next location prediction problem, thus are excluded from this study. For example *destination prediction* focuses on predicting the destination of an individual without considering the path to reach it [62, 106]. Similarly, *next stay-point prediction* predicts the next location in which an individual will stay ignoring

the intermediate locations and the interval time [110]. Finally, *path prediction with known destinations* finds the best path to reach a destination [55].

We also exclude approaches that utilize information such as speed or direction in addition to trajectories [44], or are built based on road networks [55] because they can not be compared to methods lacking such information.

6.2 Classification of Methods

We classify next location prediction approaches according to four different aspects: personalization, temporal representation, spatial representation, and their mobility learning approach.

6.2.1 Personalization

Location prediction approaches utilize the past trajectories of a moving object to predict its future locations, however, these methods differ whether they use the history/trajectories of other objects additionally. With this regard, we categorize location prediction methods in literature into two groups, i) *Individual-based* methods assume the mobility behavior of individuals are independent from each other and only utilize the history of one object to predict its future locations [43, 81, 108], while ii) general-based methods use the movement history of other objects additionally (e.g. similar objects or similar trajectories) to predict the object's future location [73, 75, 110, 114].

We evaluate individual-based and general-based methods using both sparse and abundant data which are collected with different sampling rates. Thus, we can identify the strengths and weaknesses of methods of each category in various settings.

6.2.2 Temporal Representation

We divide location prediction methods based on whether they incorporate time in their trajectory representation into three categories:

i) Location-series (No-time) representations define trajectories as a set of sequenced locations ordered in time [92]. Since our problem definition requires a fixed-interval time between two consecutive locations, we only consider the methods from this category that hold the same assumption. As mentioned in Section 6.1, next stay-point prediction do not consider the interval time or intermediate locations, thus such methods are excluded.

ii) Fixed-interval time representations use a fixed time interval between two consecutive locations [92, 114], therefore, there might be missing locations at some time points. Missing locations are generally interpolated [114]. If there are more than one location per interval, one location is chosen randomly [88] or an aggregated location is used. The choice of an appropriate time interval is crucial for the accuracy and usability of these methods. In our experiments, we use a variety of different interval time to further evaluate these methods.

iii) Variable-interval time representations allow variable transition times between sequenced locations [73], therefore, no location interpolation is required. By choosing proper Δt value, we are able to compare these methods to other methods that use fixed-interval times.

It is noteworthy, while some methods benefit from periodic and seasonal information such as time-of-day, day-of-week, or month [43, 70, 81], other works ignore such information [67]. We will investigate the effect of such information on performance of methods in our experiments.

6.2.3 Spatial Reperesentation

Location prediction approaches can be categorized based on how they represent locations in trajectories. Real datasets include raw trajectories of continuous GPS

coordinates which are noisy and inaccurate, therefore different methods are used to extract meaningful discrete locations from raw data.

- Mining frequent/dense regions using clustering methods is one of the most common ways to represent locations. Popular clustering algorithms include density-based algorithms such as DBSCAN [43] and hierarchical clustering [114]. Variants of the k-means algorithm is also used for location clustering [9].
- Grid-based methods divide space into fixed-size cells which can be simple rectangular regions [63] or use the hierarchical triangular mesh approach to divide the Earths surface into a set of triangular regions, each roughly occupying an equal area of the Earth [70].
- Semantic-based methods use semantic features of locations in addition to the geographic information, therefore, trajectories consist of meaningful locations (e.g. home, bank, school) [5, 110].

6.2.4 Learning Mobility Behavior

Location prediction approaches use different schemes to learn mobility behavior of moving objects. *Pattern-based* methods use pattern mining algorithms to extract interesting/frequent individual or group movement patterns form trajectories [5, 73, 75, 81]. *Model-based* methods formulate the movement of moving objects using mathematical models [9, 67, 92, 114].

Pattern-based prediction

Pattern-based prediction methods exploit pattern mining algorithms for next location prediction. Here, we briefly review related work in this category. Sequential pattern mining approaches extract frequent patterns and association rules from trajectories with fixed-interval time. These association rules are then matched based on the notion of support and confidence to predict the future locations of a moving

object [108]. Morzy et. al [74] reports an accuracy of 80% for next location prediction using a modified version of Apriori [3]. PrefixSpan [78] algorithm is also used to discover frequent trajectory patterns [75, 110].

WhereNext [73] first extracts *Trajectory Patterns* [36] from historical trajectories of moving objects. The *Trajectory Patterns* represent the behaviors of moving objects as sequences of regions frequently visited with a typical travel time. Then a prefix tree called *T-pattern Tree* is built where the nodes represent the frequently visited regions in the *Trajectory Patterns* and the edges represent travels among regions annotated with the typical travel time. For each input trajectory, WhereNext computes a score associated with each path in the *T-pattern Tree* and outputs the predictions based on the paths with the highest score.

A non-parametric model is proposed in [81] which predicts locations in far future. The location prediction is formulated as a eigendecomposition problem to extract patterns and associate them to temporal aspects such as day of week.

Model-based Prediction

Model-based prediction methods formulate the movement of moving objects using mathematical models. Here, we briefly review the related work in this category.

State-Space models are suitable for estimating near future locations of moving objects in a discrete location space [9,67,114]. *Markov Chain* is the simplest state-space model which is frequently used for next location prediction [9]. Using cellular network data, Lu. et. al. recently showed that Markov Chain models can result in high accuracy for trajectory prediction, which is close to theoretical upper limits of predictability for human mobility [67].

Hidden Markov Models are also used for next location prediction with an individual's activities, goals, or motivations as hidden variables [70, 114]. A location prediction approach based on *Mixed Markov Chain Models* is proposed in [8] where an individual's personality is modeled as a hidden variable.

Another approach builds a recursive model by expressing the next movement of

Table 6.1: Location prediction algorithms

Year	Name	Personalization	Spatial Representation	Temporal Representation	Mobility Scheme
Baseline	Markov Chain	General/Individual-based	Grid-based	Fixed-interval	Model-based
2003	RMF	Individual-based	Grid-based	Fixed-interval	Model-based
2008	Hybrid	Individual-based	Clustering-based	Fixed-interval	Hybrid
2013	Semi-Lazy	General-based	Clustering-based	Fixed-interval	Model-based

individuals as a recursive function of their past locations [92].

Hybrid Prediction

A hybrid method is proposed in [43] which includes a combination of model-based and pattern-based approach. Recursive motion function is adopted as the model-based part for near future predictions. A sequential pattern mining algorithm based on Apriori algorithm is also used to overcome the shortcomings of RMF.

6.3 Location Prediction Algorithms

In this section, we review three state-of-the-art location prediction methods along with a baseline method based on Markov Chains which we will later evaluate empirically. Table 6.1 provides a chronological summary of these methods classified using the aspects discussed in Section 6.2. We follow the same order of methods in the subsequent sections.

6.3.1 Markov Chain Models

We use Markov Chain (MC) based models as our baseline methods in our evaluations. Ashbrook et. al. argued that the order of the Markov Chain model could be selected based on availability of data, which limited their models to 1st and 2nd order models [9]. However, a recent study showed that higher order models did not necessarily improve the prediction accuracy as compared to a 1st order model [67].

In our study, we implemented different order Markov Chain models shown as MC(n) with n as the order. One problem with higher order MC models is lack of matching trajectories with the length of n in the database. To avoid such problem, we use MC($n-1$) for prediction when there is no matching historical data for MC(n). For spatial representation, we used a grid-based method to discretize raw locations into $g \times g$ fixed-size rectangle cells.

6.3.2 Recursive Motion Function

The work in [92] is based on the assumption that the future movement of individuals can be expressed as a recursive function of their previous locations rather than a closed formula based on time. Therefore, having the recent locations of an individual, they predict her near future movements by building a *recursive motion function* (RMF) which adapts itself as the individual moves. RMF method can be summarized in two steps as below.

Step 1: Build a recursive motion function For a partial trajectory of a moving object o at current time point t_c denoted as $\langle\langle x_1, y_1\rangle, ..., \langle x_{c-1}, y_{c-1}\rangle, \langle x_c, y_c\rangle\rangle$, the motion function can be described as Equation 6.1.

$$\begin{aligned} \langle x_{c+1}, y_{c+1}\rangle = C_0\langle x_c, y_c\rangle \\ + C_1\langle x_{c-1}, y_{c-1}\rangle + \\ + C_{f-1}\langle x_{c-f+1}, y_{c-f+1}\rangle \end{aligned} \tag{6.1}$$

where f is the retrospect which means f recent locations are considered in the function and $C_i, 0 \leq i \leq f-1$ are 2×2 matrices ($2D$ location space).

To solve this equation, they propose a motion state $s_t^{(o)}$ as a vector of f recent locations of the object o at time t as
$s_{c+1}^{(o)} = \{x_{c+1}, y_{c+1}, x_c, y_c, ..., x_{c-f+1}, y_{c-f+1}\}$ and then simplify the Equation 6.1 as Equation 6.2.

$$s_{c+1}^{(o)} = K^{(o)} s_c^{(o)} \tag{6.2}$$

where $K^{(o)}$ is the constant $(2f) \times (2f)$ motion matrix for the object. Note that the values of this matrix is only unknown for the first 2 rows which determine x_{c+1} and y_{c+1}. This equation is then solved for each row of $K^{(o)}$ separately using linear equation solving methods. The authors observe that all objects that follow the same motion pattern have identical motion matrices.

Step 2: Exploit the motion function for prediction Having $K^{(o)}$, future movements of object o is predicted using the Equation 6.2. They evaluate their method using small sets of synthetic data which are generated using predefined mathematical functions with both known motion matrices (i.e. polynomial, sinusoid, circle, and ellipse) and unknown motion matrices (i.e. spiral, peach, parabola, and swirl). Their results show that RMF outperforms a simple linear prediction model.

For our evaluation purpose, we use grid-based method for location descritizing. We adopt the same parameter settings used for Markov Chain models as described in Section 6.3.1.

6.3.3 Hybrid

The selected hybrid prediction method for moving objects [43] – which is called hybrid method throughout the chapter– predicts an objects future locations based on its historical pattern information and motion functions that are computed by recent movements. For objects trajectory patterns, they first use the DBSCAN clustering method to detect frequent regions and then apply the Apriori algorithm to find frequent patterns. Specifically, they use a novel indexing and access method for efficient query processing which is called Trajectory Pattern Tree (TPT). TPT is a variant of Signature tree [68], which is a dynamic balanced tree and specifically designed for bitmaps. Each leaf node of the TPT contains entries of encoded pattern key for each trajectory pattern, corresponding confidence and the consequence of the pattern. The insert, delete and split operations are similar to those in signature tree

Table 6.2: Datasets

Dataset	No. of objects	Time unit	Duration	Max. missing rate	No. of test objects	Prediction window size
PSyn	1000	1 step	480 steps	0.0	100	48 steps
BF	10K	1 step	500 steps	0.0	200	50 steps
GL	22	1 hour	7 days	0.7	22	1 day
TD	3000	5 min	6 days	0.5	100	1 day

and R-tree. In addition, they use two query processing methods for both non-distant and distant time queries to improve the accuracy of predictions.

6.3.4 Semi-Lazy

A semi-lazy approach for probabilistic path prediction with unknown destination is suggested in [114] for moving objects in a dynamic environment. In this approach, historical trajectories are kept and indexed for a short period of time (e.g. an hour) in a grid-based data structure. To perform prediction for a target object, they match its past trajectory against historical trajectories and extract a small set of reference trajectories using a distance-based similarity measure (Finding trajectories with locations that could be mapped to every location point of current trajectory in the same order of appearance. Then a local Hidden Markov Model is built on top of these trajectories. Circular shaped states are mined using a hierarchical clustering method applied on reference points at each time stamp. To choose the best cluster sizes, a heuristic method is suggested to choose the optimal compromise between the transition probabilities and radius of the clusters. State transition probability is then calculated as a function of intersection of states with the reference objects. This model is then used to predict the future movement of the target object with a self-correcting feature which assigns credits to reference objects based on the quality of their prediction (after actual paths are observed). These credits are incorporated into the clustering and prediction methods to improve the object's future predictions.

6.4 Experiments

6.4.1 Experiment setup

Datasets

We evaluate the performance of the prediction algorithms on two real-world and two synthetic datasets. The datasets are chosen carefully by including different sampling and missing location rates to facilitate the comparison of selected approaches. Table 6.2 summaries their details.

PSyn We generated a periodic synthetic dataset (PSyn) based on the method proposed in [69]. PSyn includes 1000 trajectories which are randomly generated from 4 different patterns of length 24 (i.e. 24 is the period at which data follows a pattern). Each trajectory includes 480 location points with no missing locations.

BF We used a modified version of Brinkhoff generator [14] to generate a synthetic set of trajectories on the road map of Boston [19]. We generated 10K moving objects with trajectories of length 500 during 500 time steps, thus each object has one location point per time step (i.e. the dataset has no missing location information).

GL Geolife GPS trajectory dataset [113] was collected in (Microsoft Research Asia) in a period of over three years. Due to sparsity of the available sample, we selected the trajectories of 22 individuals in a period of one week with missing location rate of less than 70% per individual. We interpolated the missing locations, so that there is a location point every an hour.

TD The T-Drive dataset [111, 112] contains trajectories of taxis within Beijing, China. We selected 6 days of data including 3K taxis with missing location rate of less than 50%. Again, we interpolated missing locations to have a location point every 5 minutes.

Evaluation Criteria

We used several metrics to evaluate the performance of prediction algorithms as follows.

- Prediction Rate: We evaluated the prediction ability of the algorithms by measuring the rate at which they could produce a prediction result. Prediction rate measures the number of predictable trajectories divided by the total number of test trajectories.

- Distance Error: We also evaluated the performance of algorithms by measuring the euclidean distance between the actual and predicted locations. We take the average of distances over all predictable trajectories as distance error.

- Prediction Accuracy: The accuracy of the algorithms are measured as the rate of correct predictions. Since different methods use different sizes and shapes of discretized regions or no regions at all, we use fixed-size grid cells to evaluate the accuracy. A prediction is considered accurate when it belongs to the same grid cell as the actual location. Prediction accuracy is measured by dividing the number of accurate predictions by the number of predictable trajectories.

  In addition to distance error and prediction accuracy, we defined two other metrics *Penalized Error* and *Penalized Accuracy* which incorporate prediction rate.

- Penalized Distance Error: We penalize a model when no prediction is possible by choosing a random location point from history as prediction. We then compute the distance between predictions/random predictions and the actual locations and divide it by the total number of test trajectories as penalized distance error.

- Penalized Prediction Accuracy: Similar to penalized error, we define penalized accuracy as accuracy when models are penalized for the lack of prediction.

Table 6.3: Parameters

Parameter	Values
g	8,16,32,64,128,256,512,1024,2048
density	1-10
n-back	1 - 7
supp/conf	0.1-0.5

Table 6.4: Tuned parameters

Method	g	density	n-back	supp/conf
MC-I	8,8,32,32	-	1	-
MC-G	32,64,16,256	-	1	-
RMF	-	-	3,3,2,2	-
Hybrid	64,64,128,32	2,4,4,7	3,3,2,2	0.3/0.3
Semi-Lazy	512,2048,128,512	-	2	-

In addition to comparing the accuracy of methods with each-other, we also used an upper limit of predictability which computes a theoretical maximal accuracy for each dataset. In this way, we can also see how well the methods are performing compared to a maximal possible accuracy.

Tuning Parameters

We tuned the parameters of each method for their best performance using our data. For location representation, the granularity of grid cells g is used to divide the map into $g \times g$ rectangles when fixed-size grids are utilized. For methods with frequent region mining approaches, g is used to choose the minimum radius of frequent regions as a factor of diagonal of the map. Another parameter *density* is used to specify the minimum number of points/trajectories in frequent region mining algorithms.

Moreover, all model-based methods use the number of backward location points *n-back* for tuning. Several methods use minimum support *supp* and confidence *conf* parameters. In addition to these parameters, each method also uses other parameters specific to it which are also tuned for each dataset. Table 6.3 shows the common parameters used in several methods and the value ranges used for tuning. Table 6.4 shows the summary of tuned parameters for each method/dataset pair. Four values shown for each parameter represent the best tuned value in each dataset in the order of PSyn, BT, GL, and TD.

Methodology

In all of our experiments, first we selected a random sample of individuals as the test sample. Then we used an incremental training window method to repeatedly increase the size of training period by incorporating the temporal aspects of trajectories. The size of test data and the prediction window in every dataset is shown in Table 6.2. For general-based methods, in each dataset, we started with the first window of all data for training and predicted a location in the next window for the individuals in the test data. Then, we increased the training window repeatedly during several steps. Final results were calculated as the average of all steps. For individual-based methods, training data only included the past history of each test individual.

6.4.2 Predictability Analysis

Methods

We used the measures proposed in [88] to compute the upper limits of predictability for each object's mobility in our datasets. Regardless of the time and location granularity in each dataset, we assume the complete history of object i as a single trajectory T^i with fixed time units for the total duration of the dataset. If an object's location is unknown for any time stamp, we mark it as 'missing'. For example, each object's history is presented by a trajectory of length 7×24 in GL dataset while

some of the locations are 'missing'. The main idea behind predictability analysis is that uncertainty (or randomness) of the trajectories can be measured by entropy with larger value indicating less predictability. We follow the notations in [88] to calculate the trajectory entropy of object i with L_i distinct locations.

S^i_{unc} is the temporally-uncorrelated entropy with p^i_k as the probability of visiting location k by object i:

$$S^i_{unc} = -\Sigma_{k=1}^{L_i} p^i_k \log_2 p^i_k$$

S^i_{real} is the real entropy considering both temporal and spatial patterns with $p(T')$ as the probability of finding the sub-trajectory T' in T^i:

$$S^i_{real} = -\Sigma_{T' \subset T^i} p^i(T') \log_2[p^i(T')]$$

Similar to [88], we estimate S^i_{real} using Lempel-Ziv data compression [56]. Clearly, $0 \leq S^i_{real} \leq S^i_{unc} \leq \infty$.

The predictability of object i's mobility Π_i at time n is bounded as $\Pi_i \leq \Pi^{max}_i$ where Π^{max}_i is calculated using Fanno's inequality [24]:

$$S^i_{real} = H(\Pi^{max}_i) + (1 - \Pi^{max}_i) log_2(L_i - 1) \tag{6.3}$$

where

$$H(\Pi^{max}_i) = -\Pi^{max}_i log_2(\Pi^{max}_i) - (1 - \Pi^{max}_i) log_2(1 - \Pi^{max}_i)$$

Similarly, Π^{unc}_i can be calculated using S^i_{unc} instead of S^i_{real}.

Predictability Limits of Our Data

We calculated the theoretical upper limits of predictability for our data which are presented in Figure 6.1. We used simple grids to divide the map into fixed-size rectangular cells. We varied the size of the cells by changing the grid resolution to see how the size of predicted regions effect the predictability. All of datasets show consistent decrease in predictability as the region size decreases. Intuitively, the next presence of individuals is more predictable in larger regions compared to more granular and

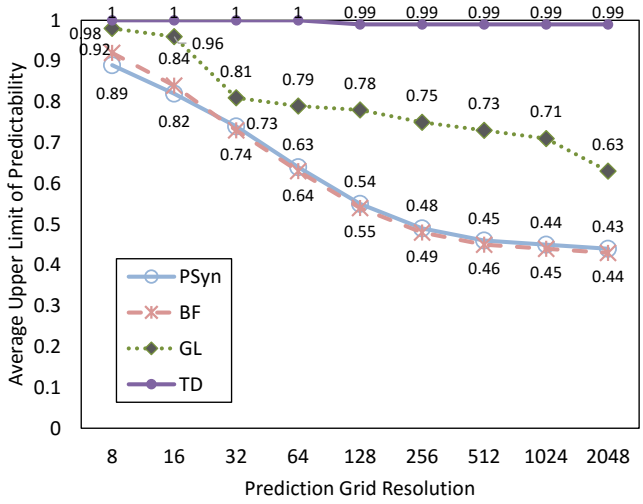


Figure 6.1: Theoretical limits of predictability in the available datasets

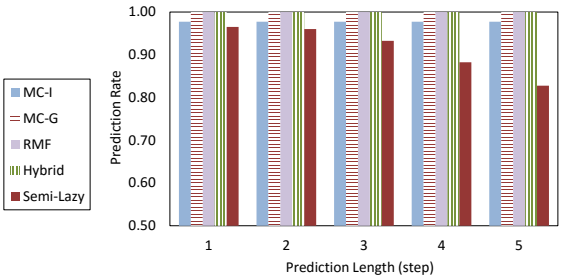
fine-grained areas. It is noteworthy, BF and PSyn include location information for every time point for all individuals while GF and TD include a minimum of 0.7% and 50% missing locations per individual respectively. The presence of missing locations are incorporated in computing the upper limits [88] which could explain some parts of the high predictability of GF and TD datasets.

6.4.3 Evaluations

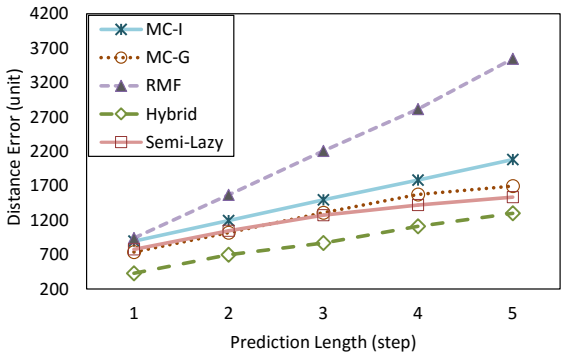
In this section, we evaluate and compare the selected methods extensively by varying the prediction length. We also compare the performance of the methods with the theoretical upper limits of predictability for each dataset. Moreover, we study the impact of the number of backward steps for model-based methods.

The Impact of the Prediction Length

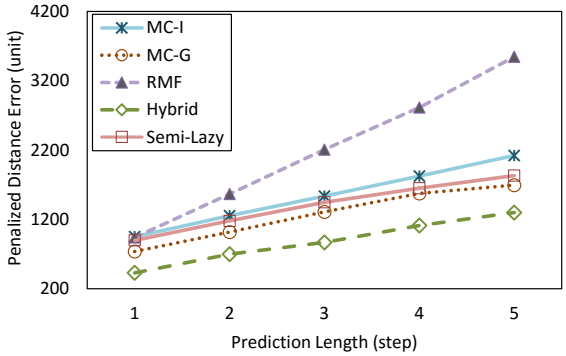
To study the performance of our algorithms in predicting the future locations, we varied prediction length and compared the prediction rate and error of methods. Figures 6.2, 6.3, 6.4, and, 6.5 show the effect of prediction length on prediction



(a) Prediction rate



(b) Distance error



(c) Penalized distance error

Figure 6.2: PSyn dataset: the impact of the prediction length

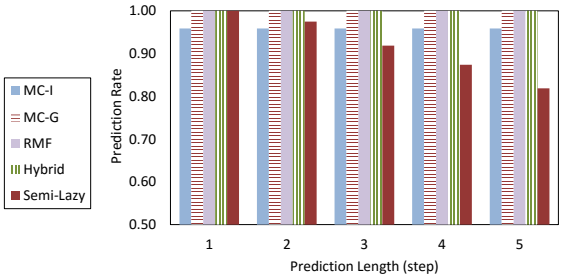
rate, distance error, and penalized distance error in different datasets. Figure. 6.2a shows the prediction rate in PSyn dataset. Semi-Lazy is the only method which is significantly affected by the length of prediction. As the prediction length is

increasing, the prediction rate is dropping for Semi-lazy while the rest of methods are not affected significantly or are not affected at all. The RMF and Hybrid methods always entail a prediction. MC-G is a general-based method using the data of all individuals with no restriction on similarity. Semi-lazy though is a general-based method, utilizes only the similar trajectories which makes it prone to the lack of prediction.

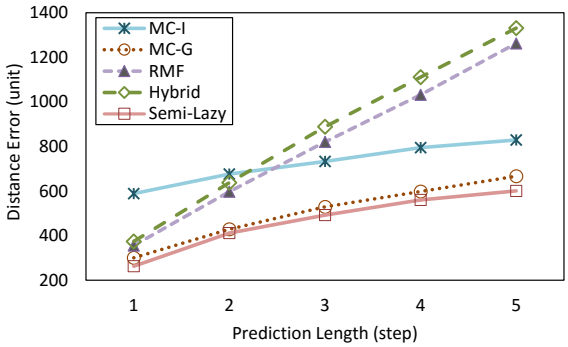
As for the distance error, the hybrid method outperforms all of the other methods for PSyn data as shown in Figure 6.2b. Semi-lazy is the next best performer resulting in a slightly lower distance error than MC-G, however, when we penalize it for the lack of prediction, MC-G outperforms it as shown in Figure 6.2c. The distance error increases as the prediction length increases for all of the methods, however, RMF's error is affected considerably higher than the other methods making it an unsuitable method for this data.

Figure 6.3a shows that the results of prediction rates for BF dataset are similar to PSyn. The distance error, on the other hand, is different as shown in Figure 6.3b. Semi-Lazy outperforms the other methods with distance error of 300 units for prediction length of one step. Again, while it results in slightly better error than MC-G, however, when penalized for the lack of prediction, it under-performs MC-G significantly as the prediction length increases in Figure 6.3c. Surprisingly, the Hybrid method performs as bad or slightly worse than RMF. It could be explained by the fact that the pattern-mining part of the Hybrid method relies heavily on periodic aspect of data while BF is not periodic.

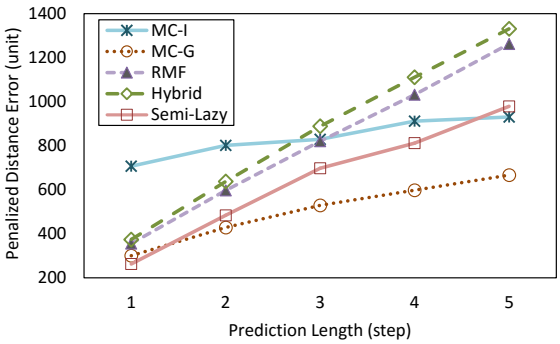
The prediction rate of Semi-Lazy drops as low as 40% for GL data in Figure 6.4a. The sampling rate of one hour in GL makes the prediction far in future rather than a near future prediction for which Semi-lazy is designed. Figure 6.4b shows that the Hybrid method performs slightly better than RMF since the data has periodic features in it. For only one step in future (i.e. the next hour in GL), the Hybrid method outperforms the other methods with distance error of 1.5km. For longer prediction lengths, Markov-based models MC-I and MC-G outperform all other methods signif-



(a) Prediction rate



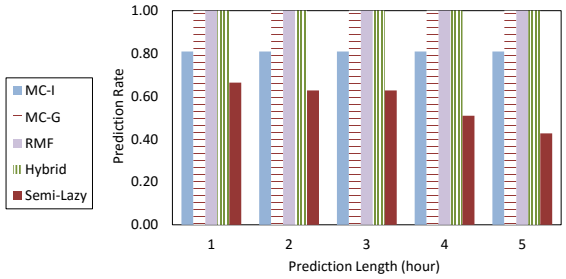
(b) Distance error



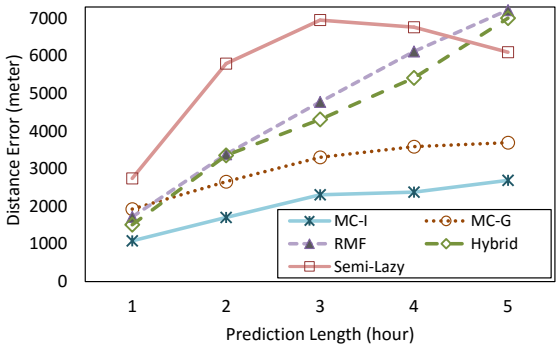
(c) Penalized distance error

Figure 6.3: BF dataset: the impact of the prediction length

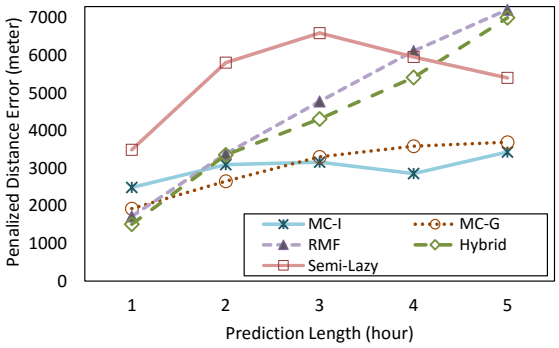
icantly. MC-I results in lower distance error without considering its low prediction rate of around 80%. After incorporating the prediction rate, we see that MC-G outperforms MC-I for the most values of prediction length. Semi-lazy constantly yields



(a) Prediction rate



(b) Distance error

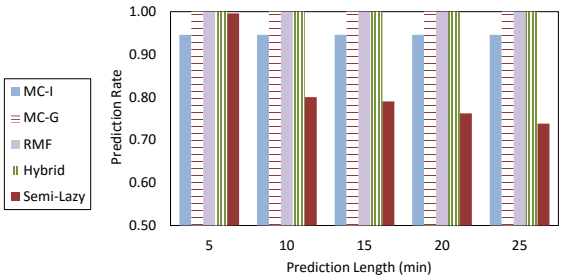


(c) Penalized distance error

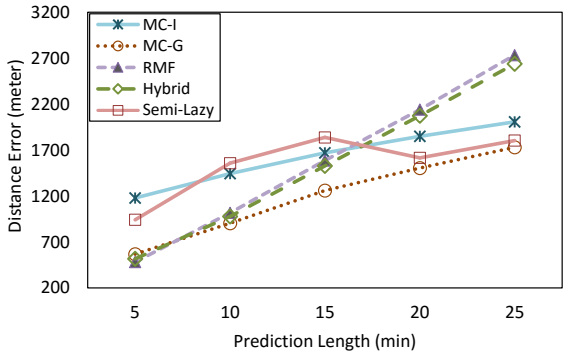
Figure 6.4: GL dataset: the impact of the prediction length

the worst error in both Figures 6.4b and 6.4c with its penalized error being slightly higher than its error as expected.

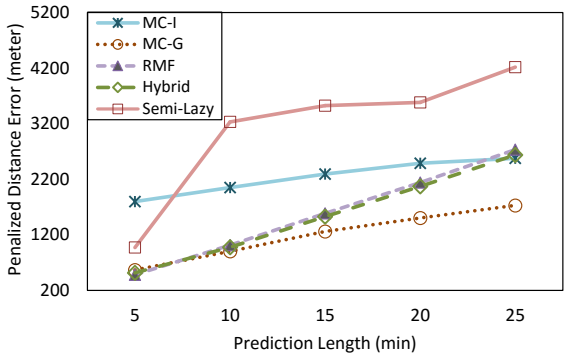
Figure 6.5a shows that TD also results in a dropping prediction rate for Semi-Lazy



(a) Prediction rate



(b) Distance error



(c) Penalized distance error

Figure 6.5: TD dataset: the impact of the prediction length

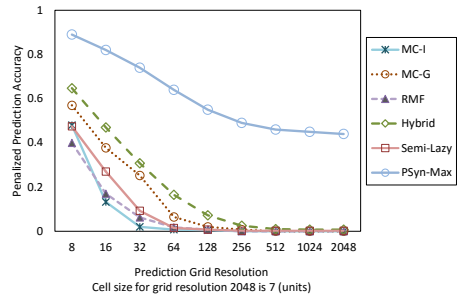
which effects its penalized distance error significantly as illustrated in Figure 6.5c. It is also noteworthy, since the data is not periodic, the hybrid method does not result in lower errors leaving MC-G as the best performer.

Comparison of Accuracy of Methods

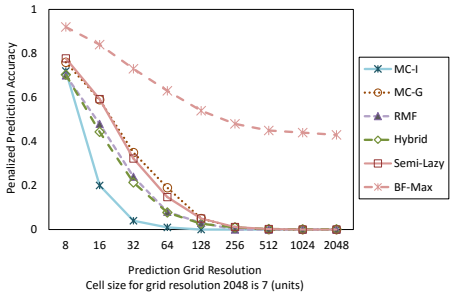
In Figure 6.6, we evaluated the accuracy of selected methods by comparing their penalized accuracy for varying prediction granularity (i.e. evaluation cell size) when prediction length is one time unit in each dataset. We also included the upper limit of predictability in each dataset to show the existing gap between the theoretical limits and the practical accuracy that could be achieved by these methods. In all of the datasets, all methods follow the same trend as the upper limits. As the size of prediction regions increases, accurate predictions are more likely. In Figure 6.6a, the Hybrid method outperforms the other methods using PSyn dataset, however, there is a considerable gap between the accuracy of methods and the upper limit which decreases as the size of prediction area increases. While the hybrid method enjoys the periodic aspect of PSyn data, it does not perform as well for BF data in Figure 6.6b in which MC-G and Semi-lazy methods yield better accuracy. There is a similar trend of decreasing gap with the upper limit as of PSyn. In both data, MC-I has the worse accuracy.

Comparing Hybrid and RMF It is also noteworthy to discuss how Hybrid and RMF methods compare in different datasets. In PSyn and GL, the hybrid method outperforms RMF as expected. Both of these datasets either are synthetically periodic as in PSyn or capture the movements of individuals in 24 hour periods as in GL. It can also be partly explained by the prediction length of one hour in GL being in favor of the hybrid method; RMF is more suitable for near future predictions. On the contrary, the hybrid method performs similar to RMF in BF and TD which means the pattern-based part of it is not performing well when no periodic data is present or the prediction length is short.

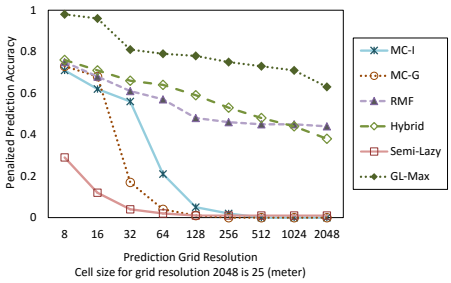
Comparing Model-based Methods MC-G performs similar or better than MC-I and Semi-Lazy in all of the datasets. Moreover, while the accuracy of all three model-based methods follow a sudden drop in real datasets in Figures 6.6c and 6.6d,



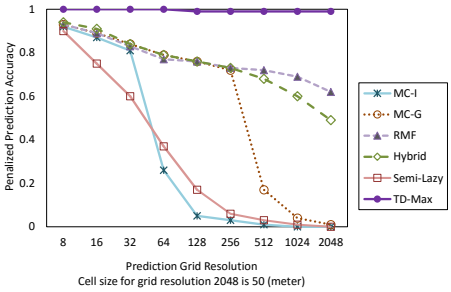
(a) PSyn dataset: prediction length is one step



(b) BF dataset: prediction length is one step



(c) GL dataset: prediction length is one hour



(d) TD dataset: prediction length is 5 minutes

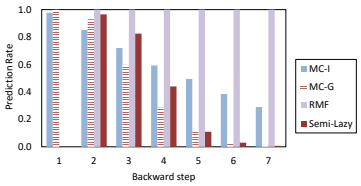
Figure 6.6: Comparison of methods predictability for different prediction granularity

MC-G shows more robustness for TD dataset. These results can be explained due to the fact that MC-G has higher prediction rate compared to MC-I and Semi-Lazy models, thus, does not suffer from penalization as much as the other two methods.

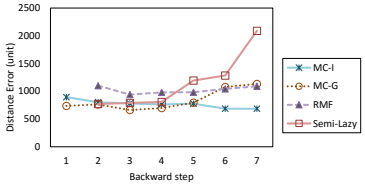
The impact of the number of backward steps

Model-based methods utilize the number of backward steps as one of their major parameters. In this section, we present the tuning results of varying this parameter and study its impact on prediction rate and error. Figure 6.7a shows how prediction rate varies for PSyn data as the number of backward steps increases. RMF always produces a prediction since it only requires the location of the backward steps and

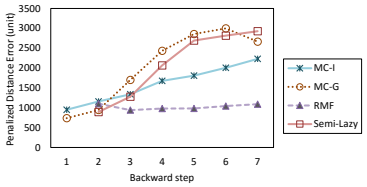
does not search for any matching or similar sequences in history. On the other hand, all of Markov-based models experience a drop in prediction rate as the number of backward steps increases. Distance error on the other hand decreases with increasing the number of steps as shown in Figure 6.7b. This indicates how methods become more precise as the number of matching locations increases. After incorporating the lack of prediction, we observe an increase in penalized error in Figure 6.7c. Similar results can be seen in other datasets.



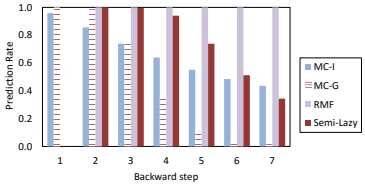
(a) PSyn dataset: prediction rate



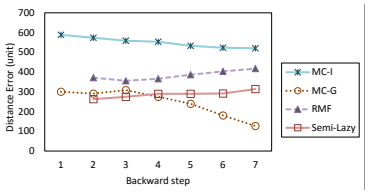
(b) PSyn dataset: distance error



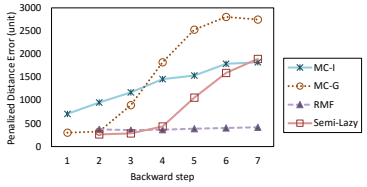
(c) PSyn dataset: penalized distance error



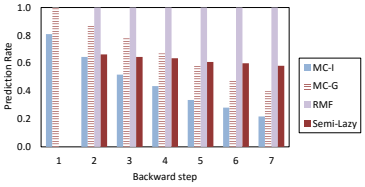
(d) BF dataset: prediction rate



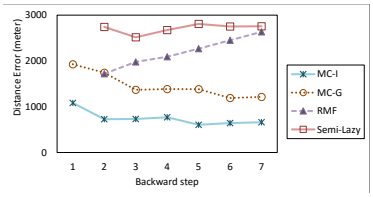
(e) BF dataset: distance error



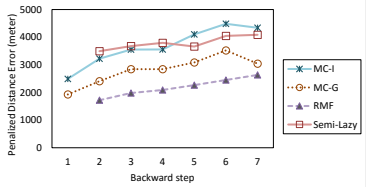
(f) BF dataset: penalized distance error



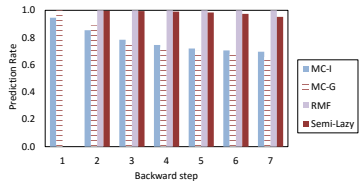
(g) GL dataset: prediction rate



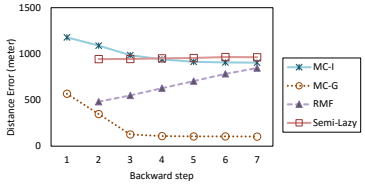
(h) GL dataset: distance error



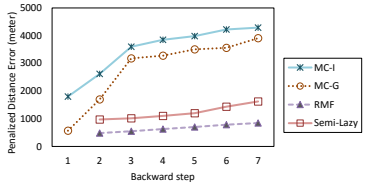
(i) GL dataset: penalized distance error



(j) TD dataset: prediction rate



(k) TD dataset: distance error



(l) TD dataset: penalized distance error

Figure 6.7: The impact of the backward steps (prediction length is 1 time unit)

Chapter 7

Conclusions and Future Work

Mobile crowd sensing (MCS) has numerous applications in a wide range of domains including syndromic surveillance, crime mapping, environmental health and pollution mapping, traffic monitoring, and emergency response. In this dissertation, we proposed methods to build robust task management frameworks to handle uncertainty and ensure privacy in such applications. Our solutions not only increase the disposition of the participants to engage in data collection and sharing activity, but also ultimately lead to more effective MCS applications.

7.1 Summary

In Chapter 3, we defined and formulated the problem of privacy-aware spatial task assignment in crowd sensing with cloaked locations as a novel two-stage optimization problem. We showed the problem to be NP-hard in each stage, and proposed efficient greedy algorithms for each stage of the optimization problem. We studied the impact of parameter values including task size, participant size, coverage goal and cloaking size on our methods and showed their effectiveness and robustness.

Chapter 4 investigated privacy-aware solutions for autonomous task selection based on the notions of private information retrieval (PIR). First, we proposed two novel solutions to answer private $RkNN$ queries without disclosing any information about the location of query point. Our solutions utilized PIR mechanisms to request data from an untrusted database server without the server learning about retrieved data or the query source. We evaluated our methods extensively using real datasets. We investigated the effect of variety of parameters on computational cost and data

storage size. Our results showed the efficiency and effectiveness of our solutions. Then, we showed that private $RkNN$ queries can be utilized by participants of MCS to query tasks autonomously without revealing their location while achieving high task coverage.

In Chapter 5, we extended the problem of spatial task assignment for dynamic MCS environments. Dynamic crowd sensing platforms come with some inherent challenges, including constantly moving participants and uncertainty (i.e. noisy or imprecise location). These were addressed in our dynamic spatial task assignment framework based on the DDDAS (Dynamic Data Driven Application Systems) paradigm. Our approach includes a data driven framework with feedback loops to steer data collection, thus adapts to moving participants and application needs in real time. Participant mobility is modeled using available public data, which is then used with an adaptive filtering module to improve uncertain locations. Finally, a spatial task assignment approach recruits the best set of participants to achieve application-specific goals such as maximum coverage or minimum cost.

We also investigated the effectiveness of existing human mobility prediction models in Chapter 6 as an attempt to make a more knowledgeable decision for our dynamic task assignment framework. To this end, we conducted an extensive experimental evaluation of several state-of-the-art location prediction methods. Our selected methods included model-based, pattern-based, and hybrid approaches to predict user locations in near or far future time. To conduct a fair and comprehensive evaluation, we carefully chose our test data sets from both real and synthetic data. We also designed extensive experiments to show both the strengths and weaknesses of each method.

7.2 Future Work

Possible future research directions for the proposed methods in this dissertation are presented as follows.

Incorporating Participant's Trust in Privacy-aware Spatial Task Assignment Privacy and trust generally follow conflicting goals, given that the participant's trust is gained by higher accuracy and exactness of provided data, but privacy aims at hiding or perturbing identifying data (e.g. using location cloaking) to protect the participant [2, 37]. Furthermore, trust issues become more challenging for anonymous tasking. Anonymity may result in assigning tasks to untrustworthy or unqualified participants [20]. In our proposed methods, we focused on privacy aspects of task assignment in which users would only share their cloaked locations. As future work, our approach could be extended to take into account the trustworthiness of participants in terms of the quality of the data contributed by them without compromising their privacy [51].

Extending The Dynamic Task Management Framework Real-world crowd sensing environments are very dynamic with constantly changing parameters. In our research, we considered moving participants as one of the core dynamic parameters of such environments. Therefore, we built a framework to adapt to constant changes of participant locations and the resulting uncertainty. As a next step, we plan to extend our approach to also consider moving tasks (i.e. targets). Examples of moving tasks include monitoring a moving crowd or traffic flow, and disease spread surveillance. Including such tasks requires building predictive models for different types of task movement patterns while feeding their changes in real-time to the task assignment process.

Bibliography

[1] Gowalla dataset, http://snap.stanford.edu/data/loc-gowalla.html.

[2] Charu C Aggarwal and Tarek Abdelzaher. Social sensing. In *Managing and Mining Sensor Data*, pages 237–297. Springer, 2013.

[3] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[4] Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, and Zahid Nawaz. Location-based crowdsourcing: extending crowdsourcing to the real world. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 13–22. ACM, 2010.

[5] Luis Otavio Alvares, Vania Bogorny, Bart Kuijpers, JAF de Macelo, B Moelans, and Andrey Tietbohl Palma. Towards semantic trajectory knowledge discovery. *Data Mining and Knowledge Discovery*, 2007.

[6] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 901–914. ACM, 2013.

[7] Claudio Agostino Ardagna, Marco Cremonini, Ernesto Damiani, S De Capitani Di Vimercati, and Pierangela Samarati. Location privacy protection through

obfuscation-based techniques. In *Data and Applications Security XXI*, pages 47–60. Springer, 2007.

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

[9] Daniel Ashbrook and Thad Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.

[10] J v Bercken, Björn Blohsfeld, Jens-Peter Dittrich, Jürgen Krämer, Tobias Schäfer, Martin Schneider, and Bernhard Seeger. Xxl-a library approach to supporting efficient implementations of advanced database queries. In *Proc. of the Conf. on Very Large Databases (VLDB)*, pages 39–48, 2001.

[11] Thomas Bernecker, Tobias Emrich, Hans-Peter Kriegel, Matthias Renz, Stefan Zankl, and Andreas Züfle. Efficient probabilistic reverse nearest neighbor query processing on uncertain data. *Proceedings of the VLDB Endowment*, 4(10):669–680, 2011.

[12] Claudio Bettini, Sushil Jajodia, and Pierangela Samarati. *Privacy in location-based applications: research issues and emerging trends*, volume 5599. Springer, 2009.

[13] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[14] Thomas Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[15] Nirupama Bulusu, Chun Tung Chou, Salil Kanhere, Yifei Dong, Shitiz Sehgal, David Sullivan, and Lupco Blazeski. Participatory sensing in commerce: Using

mobile camera phones to track market price dispersion. In *Proceedings of the International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems (UrbanSense 2008)*, pages 6–10, 2008.

[16] Xin Cao, Gao Cong, and Christian S Jensen. Mining significant semantic locations from gps data. *Proceedings of the VLDB Endowment*, 3(1-2):1009–1020, 2010.

[17] Muhammad Aamir Cheema, Xuemin Lin, Wei Wang, Wenjie Zhang, and Jian Pei. Probabilistic reverse nearest neighbor queries on uncertain data. *Knowledge and Data Engineering, IEEE Transactions on*, 22(4):550–564, 2010.

[18] Muhammad Aamir Cheema, Wenjie Zhang, Xuemin Lin, and Ying Zhang. Efficiently processing snapshot and continuous reverse k nearest neighbors queries. *The VLDB Journal*, 21(5):703–728, 2012.

[19] Su Chen, Christian S Jensen, and Dan Lin. A benchmark for evaluating moving object indexes. *Proceedings of the VLDB Endowment*, 1(2):1574–1585, 2008.

[20] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, 84(11):1928–1946, 2011.

[21] Delphine Christin. Privacy in mobile participatory sensing: Current trends and future challenges. *Journal of Systems and Software*, 2015.

[22] C. Cornelius, D. Kotz A. Kapadia, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: Privacy-aware people-centric sensing. volume 5, pages 211–224. in Proceedings of the International Conference on Mobile Systems, Applications and Services (MobiSys), ACM Press, Jun 2008.

[23] Antonio Corral, Yannis Manolopoulos, Yannis Theodoridis, and Michael Vassilakopoulos. Algorithms for processing k-closest-pair queries in spatial databases. *Data & Knowledge Engineering*, 49(1):67–104, 2004.

[24] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[25] Frederica Darema. Dynamic data driven applications systems: A new paradigm for application simulations and measurements. In *Computational Science-ICCS 2004*, pages 662–669. Springer, 2004.

[26] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. Prism: Platform for remote sensing using smartphones. pages 63–76. in Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys), 2010.

[27] Dingxiong Deng, Cyrus Shahabi, and Ugur Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 324–333. ACM, 2013.

[28] Linda Deng and Landon P Cox. Livecompare: grocery bargain hunting through participatory sensing. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, page 4. ACM, 2009.

[29] Cynthia Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.

[30] Shane B Eisenman, Emiliano Miluzzo, Nicholas D Lane, Ronald A Peterson, Gahng-Seop Ahn, and Andrew T Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):6, 2009.

[31] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. ACM, 2008.

[32] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. De-anonymization attack on geolocated data. *Journal of Computer and System Sciences*, 2014.

[33] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: Current state and future challenges. *Communications Magazine, IEEE*, 49(11):32–39, 2011.

[34] G. Ghinita. *Privacy for Location-Based Services*. Synthesis Lectures on Information Security, Privacy, and Tru. Morgan & Claypool, 2013.

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

[36] Fosca Giannotti, Mirco Nanni, and Dino Pedreschi. Efficient mining of temporally annotated sequences. In *In Proc. SDM06*. Citeseer, 2006.

[37] Peter Gilbert, Jaeyeon Jung, Kyungmin Lee, Henry Qin, Daniel Sharkey, Anmol Sheth, and Landon P Cox. Youprove: authenticity and fidelity in mobile sensing. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 176–189. ACM, 2011.

[38] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[39] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140, pages 107–113. IET, 1993.

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

[41] Jeff Howe. *Crowdsourcing: How the power of the crowd is driving the future of business*. Random House, 2008.

[42] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 125–138. ACM, 2006.

[43] Hoyoung Jeung, Qing Liu, Heng Tao Shen, and Xiaofang Zhou. A hybrid prediction model for moving objects. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 70–79. IEEE, 2008.

[44] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, and Christian S Jensen. Path prediction and predictive range querying in road network databases. *The VLDB Journal*, 19(4):585–602, 2010.

[45] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

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

[47] Ibrahim Kamel and Christos Faloutsos. Hilbert r-tree: An improved r-tree using fractals. 1993.

[48] Leyla Kazemi and Cyrus Shahabi. A privacy-aware framework for participatory sensing. *ACM SIGKDD Explorations Newsletter*, 13(1):43–51, 2011.

[49] Leyla Kazemi and Cyrus Shahabi. Towards preserving privacy in participatory sensing. In *Pervasive Computing and Communications Workshops, 2011 IEEE International Conference on*, pages 328–331, 2011.

[50] Leyla Kazemi and Cyrus Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 189–198. ACM, 2012.

[51] Leyla Kazemi, Cyrus Shahabi, and Lei Chen. Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 314–323. ACM, 2013.

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

[53] Ali Khoshgozaran, Cyrus Shahabi, and Houtan Shirani-Mehr. Location privacy: going beyond k-anonymity, cloaking and anonymizers. *Knowledge and Information Systems*, 26(3):435–465, 2011.

[54] Hidetoshi Kido, Yutaka Yanagisawa, , and Tetsuji Satoh. An anonymous communication technique using dummies for location-based services. In Proceedings of the IEEE International Conference on Pervasive Services, 2005.

[55] Sang-Wook Kim, Jung-Im Won, Jong-Dae Kim, Miyoung Shin, Junghoon Lee, and Hanil Kim. Path prediction of moving objects on road networks through analyzing past trajectories. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 379–389. Springer, 2007.

[56] Ioannis Kontoyiannis, Paul H Algoet, Yu M Suhov, and Abraham J Wyner. Nonparametric entropy estimation for stationary processes and random fields, with applications to english text. *Information Theory, IEEE Transactions on*, 44(3):1319–1327, 1998.

[57] Flip Korn and S Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *ACM SIGMOD Record*, volume 29, pages 201–212. ACM, 2000.

[58] Andreas Krause, Eric Horvitz, Aman Kansal, and Feng Zhao. Toward community sensing. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 481–492. IEEE, 2008.

[59] John Krumm. Inference attacks on location tracks. In *Pervasive Computing*, pages 127–143. Springer, 2007.

[60] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[61] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[62] John Krumm and Eric Horvitz. Predestination: Inferring destinations from partial trajectories. In *UbiComp 2006: Ubiquitous Computing*, pages 243–260. Springer, 2006.

[63] Po-Ruey Lei, Tsu-Jou Shen, Wen-Chih Peng, and Jiunn Su. Exploring spatial-temporal trajectory model for location prediction. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 58–67. IEEE, 2011.

[64] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-Hua Teng. On trip planning queries in spatial databases. In *International Symposium on Spatial and Temporal Databases*, pages 273–290. Springer, 2005.

[65] Jiajia Li, Botao Wang, and Guoren Wang. Efficient probabilistic reverse k-nearest neighbors query processing on uncertain data. In *International Conference on Database Systems for Advanced Applications*, pages 456–471. Springer, 2013.

[66] Xiang Lian and Lei Chen. Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data. *The VLDB JournalThe International Journal on Very Large Data Bases*, 18(3):787–808, 2009.

[67] Xin Lu, Erik Wetter, Nita Bharti, Andrew J Tatem, and Linus Bengtsson. Approaching the limit of predictability in human mobility. *Scientific reports*, 3, 2013.

[68] Nikos Mamoulis, David W Cheung, and Wang Lian. Similarity search in sets and categorical data using the signature tree. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 75–86. IEEE, 2003.

[69] Nikos Mamoulis, Cao Huiping, Kollios George, Hadjieleftheriou Marios, Tao Yufei, and David W. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 236–245. ACM, 2004.

[70] Wesley Mathew, Ruben Raposo, and Bruno Martins. Predicting future locations with hidden markov models. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 911–918. ACM, 2012.

[71] Harvey J Miller and Jiawei Han. *Geographic data mining and knowledge discovery*. CRC Press, 2009.

[72] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 323–336. ACM, 2008.

[73] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–646. ACM, 2009.

[74] Mikołaj Morzy. Prediction of moving object location based on frequent trajectories. In *Computer and Information Sciences–ISCIS 2006*, pages 583–592. Springer, 2006.

[75] Mikołaj Morzy. Mining frequent trajectories of moving objects for location prediction. In *Machine Learning and Data Mining in Pattern Recognition*, pages 667–680. Springer, 2007.

[76] Kyriakos Mouratidis. Strong location privacy: A case study on shortest path queries. In *Data Engineering Workshops (ICDEW), 2013 IEEE 29th International Conference on*, pages 136–143. IEEE, 2013.

[77] Stavros Papadopoulos, Spiridon Bakiras, and Dimitris Papadias. Nearest neighbor search with strong location privacy. *Proceedings of the VLDB Endowment*, 3(1-2):619–629, 2010.

[78] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *icccn*, page 0215. IEEE, 2001.

[79] S. Reddy, D. Estrin, and M. Srivastava. Recruitment framework for participatory sensing data collections. pages 138–155. in Proceedings of the 8th International Conference on Pervasive Computing, Springer Berlin Heidelberg, May 2010.

[80] Mehdi Riahi, Thanasis G. Papaioannou, Immanuel Trummer, and Karl Aberer. Utility-driven data acquisition in participatory sensing. EDBT/ICDT, ACM, March 2013.

[81] Adam Sadilek and John Krumm. Far out: Predicting long-term human mobility. In *AAAI*, 2012.

[82] Hans Sagan. Hilberts space-filling curve. In *Space-Filling Curves*, pages 9–30. Springer, 1994.

[83] Mehdi Sharifzadeh and Cyrus Shahabi. Vor-tree: R-trees with voronoi diagrams for efficient processing of spatial nearest neighbor queries. *Proceedings of the VLDB Endowment*, 3(1-2):1231–1242, 2010.

[84] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos. Anonysense: A system for anonymous opportunistic sensing. *Journal of Pervasive and Mobile Computing*, 7(1):16–30, 2010.

[85] Houtan Shirani-Mehr, Farnoush Banaei-Kashani, and Cyrus Shahabi. Efficient viewpoint assignment for urban texture documentation. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 62–71. ACM, 2009.

[86] Reza Shokri, George Theodorakopoulos, J Le Boudec, and J Hubaux. Quantifying location privacy. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 247–262. IEEE, 2011.

[87] Petr Slavík. Improved performance of the greedy algorithm for partial cover. *Information Processing Letters*, 64(5):251–254, 1997.

[88] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[89] Ioana Stanoi, Divyakant Agrawal, and Amr El Abbadi. Reverse nearest neighbor queries for dynamic databases. In *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, pages 44–53, 2000.

[90] Girts Strazdins, Artis Mednis, Georgijs Kanonirs, Reinholds Zviedris, and Leo Selavo. Towards vehicular sensor networks with android smartphones for road surface monitoring. In *2nd International Workshop on Networks of Cooperating Objects (CONET11), Electronic Proceedings of CPS Week*, volume 11, 2011.

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

[92] Yufei Tao, Christos Faloutsos, Dimitris Papadias, and Bin Liu. Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings*

of the 2004 ACM SIGMOD international conference on Management of data, pages 611–622. ACM, 2004.

[93] Yufei Tao, Dimitris Papadias, and Xiang Lian. Reverse knn search in arbitrary dimensionality. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 744–755. VLDB Endowment, 2004.

[94] Yufei Tao, Dimitris Papadias, Xiang Lian, and Xiaokui Xiao. Multidimensional reverse knn search. *The VLDB Journal*, 16(3):293–316, 2007.

[95] Hien To, Gabriel Ghinita, and Cyrus Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *Proceedings of the VLDB Endowment*, 7(10), 2014.

[96] Vijay V. Vazirani. *Approximation algorithms*. 2001.

[97] Khuong Vu, Rong Zheng, and Lie Gao. Efficient algorithms for k-anonymous location privacy in participatory sensing. In *INFOCOM, 2012 Proceedings IEEE*, pages 2399–2407, 2012.

[98] Lu Wang, Ruxia Ma, and Xiaofeng Meng. Evaluating k nearest neighbor query on road networks with no information leakage. In *Web Information Systems Engineering–WISE 2015*, pages 508–521. Springer, 2015.

[99] Lu Wang, Xiaofeng Meng, Haibo Hu, and Jianliang Xu. Bichromatic reverse nearest neighbor query without information leakage. In *Database Systems for Advanced Applications*, pages 609–624. Springer, 2015.

[100] Shengsheng Wang, Yang Li, Sheng Chai, and Bolou Dickson Bolou. Sphlu: An efficient algorithm for processing prknn queries on uncertain data. *Chinese Journal of Electronics*, 25(3):403–406, 2016.

[101] Song Wang and X Sean Wang. In-device spatial cloaking for mobile user privacy assisted by the cloud. In *Mobile Data Management, 2010 Eleventh International Conference on*, pages 381–386. IEEE, 2010.

[102] Yijie Wang, Xiaoyong Li, Xiaoling Li, and Yuan Wang. A survey of queries over uncertain data. *Knowledge and Information Systems*, pages 1–46, 2013.

[103] Peter Williams and Radu Sion. Usable pir. In *NDSS*, 2008.

[104] Wei Wu, Fei Yang, Chee-Yong Chan, and Kian-Lee Tan. Finch: Evaluating reverse k-nearest-neighbor queries on location data. *Proceedings of the VLDB Endowment*, 1(1):1056–1067, 2008.

[105] Yonghui Xiao and Li Xiong. Protecting locations with differential privacy under temporal correlations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1298–1309, 2015.

[106] Andy Yuan Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 254–265. IEEE, 2013.

[107] Shiyu Yang, Muhammad Aamir Cheema, Xuemin Lin, and Ying Zhang. Slice: Reviving regions-based pruning for reverse k nearest neighbors queries. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 760–771. IEEE, 2014.

[108] Gökhan Yavaş, Dimitrios Katsaros, Özgür Ulusoy, and Yannis Manolopoulos. A data mining approach for location prediction in mobile environments. *Data & Knowledge Engineering*, 54(2):121–146, 2005.

[109] Yang Ye, Yu Zheng, Yukun Chen, Jianhua Feng, and Xing Xie. Mining individual life pattern based on location history. In *Mobile Data Management: Systems, Services and Middleware, 2009. Tenth International Conference on*, pages 1–10. IEEE, 2009.

[110] Josh Jia-Ching Ying, Wang-Chien Lee, Tz-Chiao Weng, and Vincent S Tseng. Semantic trajectory mining for location prediction. In *Proceedings of the 19th*

ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pages 34–43. ACM, 2011.

[111] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324. ACM, 2011.

[112] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108. ACM, 2010.

[113] Yu Zheng, Xing Xie, and Wei-Ying Ma. Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.

[114] Jingbo Zhou, Anthony KH Tung, Wei Wu, and Wee Siong Ng. A semi-lazy approach to probabilistic path prediction. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 748–756. ACM, 2013.