Towards Effective Spatial Data Mining: Uncertainty, Condensity and Privacy

Bozhong Liu

Faculty of Engineering and Information Technology

University of Technology, Sydney

A thesis submitted for the degree of

Doctor of Philosophy

April 2017

Certificate of Original Authorship

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Student: Bozhong Liu

Date: 04/21/2017

Acknowledgements

Firstly, I would like to express my sincere gratitude to my principal supervisor, Dr. Ling Chen, for her continuous support of my Ph.D study and related research, for her patience, kindly support and inspiring motivation. She is always supportive when I feel frustrated or despairing. Her guidance helps me in all the time of my research life. Also, I am very thankful to Prof. Zhu, for his immense knowledge and wonderful advices. Without their guidance, my research life would be much more difficult. I am also grateful to my supervisor in Shanghai Jiao Tong University, Prof. Qiu, for his grate support and encouragement. Without his effort, I might have lost this precious opportunity of studying in UTS.

Secondly, I would like to thank my fellow students for their suggestion, discussion, cooperation and of course friendship. Their patience and support help me in overcoming numerous obstacles I have been facing through my research. Especially, I am grateful to Chunyang Liu, Meng Fang, Zhe Xu, Zhibin Hong and Shirui Pan for their kindness and sincerity. They not only help me a lot when I came to Sydney, but also provide many nice advices for my research work. They make my Ph.D study more colorful and wonderful.

Last but not the least, I would like to thank my parents for supporting me spiritually throughout writing this thesis and my life in general.

Abstract

Spatial data mining (SDM) is a process of knowledge discovery that the observing data is related to geographical information. It has become an important data mining task due to the explosive growth and pervasive use of spatial data. It is more difficult to extract interesting and useful patterns from spatial datasets due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation. Although existing methods can handle the spatial mining task properly, as the arrival of the big data era, new challenges for SDM are arising.

Firstly, traditional SDM methods usually focus on deterministic datasets, where spatial events occur affirmatively at precise locations. However, the inherent uncertainty of spatial data makes the mining process more difficult. Classical spatial data mining algorithms are no longer applicable or need delicate modification. Secondly, traditional SDM frameworks produce an exponential number of patterns, which makes it hard for users to understand or apply. To solve the condensity issue, novel techniques such as summarization or representation must be carefully investigated. Thirdly, spatial data usually involves an individual's location information, which incurs location privacy problem. It would be a challenge to protect location privacy with enhanced data security and improved resulting accuracy.

To address the uncertainty issue, we study the problem of discovering co-location patterns in the context of continuously distributed uncertain data, namely Probabilistic Co-location Patterns Mining (PCPM). We develop an effective probabilistic co-location mining framework integrated with optimization strategies to address the challenges.

To address the condensity issue, we investigate the problem of Representative Co-location Patterns Mining (RCPM). We define a new measure to quantify the distance between co-location patterns, and develop two efficient algorithms for summarization.

To address the privacy issue, we solve the problem of protecting Location Privacy in Spatial Crowdsourcing (LPSC). We propose a secure spatial crowdsourcing framework based on encryption, and devise a novel secure indexing technique for efficient querying.

The experimental results demonstrate the effectiveness and efficiency of our proposed solutions. The methods and techniques used in solving concrete SDM tasks can also be applied or extended to other SDM scenarios.

Contents

1 Introduction **1**

2 Mining Co-location Patterns from Uncertain Data **7**

2.1 Introduction 8
2.2 Related Works 10
2.3 Problem Definitions 11
2.3.1 Co-location Patterns in Deterministic Data 11
2.3.2 Co-location Patterns in Gaussian-based Data 13
2.4 Probabilistic Participation Ratio Computation 15
2.5 Probabilistic Co-location Mining Framework 18
2.6 Finding Probabilistic Neighbors 20
2.6.1 Minimum Bounding Sphere 21
2.6.2 The filtering 22
2.7 Performance Study 23
2.7.1 Experiment Setup 23
2.7.2 Comparisons with other methods 24
2.7.3 Efficiency of Filtering 26
2.7.4 Parameter Evaluation 27
2.8 Conclusion 28

3 Summarizing Spatial Co-location Patterns **29**

3.1 Introduction 30
3.2 Related Works 33
3.3 Preliminary 35
3.3.1 Co-location Patterns 35
3.3.2 Co-location Distance Measure 36

3.3.3 Problem Statement . . . 38
3.4 The *RCPFast* Algorithm . . . 40
3.5 The *RCPMS* Algorithm . . . 42
3.5.1 Optimization Strategy . . . 45
3.5.2 Approximation Strategy . . . 49
3.5.3 The *gen_cover_set()* Function . . . 51
3.6 Experimental Study . . . 52
3.6.1 Experiments on Synthetic Data . . . 52
3.6.2 Experiments on Real Data . . . 59
3.7 Conclusions . . . 62

4 Protecting Location Privacy in Spatial Crowdsourcing 63
4.1 Introduction . . . 64
4.2 Related Works . . . 66
4.2.1 Location Privacy . . . 66
4.2.2 Secure Index . . . 67
4.2.3 Task Assignment in SC . . . 68
4.3 Preliminariy . . . 69
4.3.1 Spatial Crowdsourcing Model . . . 69
4.3.2 Threat Model . . . 70
4.3.3 Paillier Cryptosystem . . . 70
4.4 The HESI Framework . . . 71
4.4.1 The Dual-Server Architecture . . . 71
4.4.2 The System Workflow . . . 72
4.5 Secure Distance Computation . . . 75
4.6 Secure Indexing . . . 78
4.6.1 SKD-tree . . . 78
4.6.2 Fast Pruning . . . 82
4.7 Secure Task Assignment . . . 86
4.7.1 Assignment Strategy . . . 86
4.7.2 Secure Assignment . . . 89
4.8 Analysis . . . 90
4.8.1 Security Analysis . . . 91
4.8.2 Complexity Analysis . . . 93

4.9 Performance Evaluation 95
4.9.1 Benchmark Data 95
4.9.2 Experimental Results 96
4.10 Conclusions 101

5 Conclusion **103**

References **105**

List of Tables

1.1 Relationships among non-spatial and spatial data [1]. 2

2.1 *Determinization* method vs. our method on EPA data. 26
2.2 *Discretization* method vs. our method on ITF data. 27

3.1 Prevalent patterns in the example. 32
3.2 Parameters used in synthetic data generation. 52

4.1 The outline of the secure protocols. 75
4.2 Potential assignments for each task. 88
4.3 Complexity summary. 94
4.4 Performance of distance computation for different location distribution. 99
4.5 Communication cost. 100
4.6 Task assignment evaluation on Yelp. 101
4.7 Task assignment evaluation on Gowalla. 101

List of Figures

2.1 An example of deterministic spatial data. 12
2.2 An example of an uncertain spatial data. 16
2.3 Using Minimum Bounding Spheres to bound ρ-regions. 22
2.4 Evaluation of filtering technique. 27
2.5 Parameter evaluation. 28

3.1 A motivating example. 31
3.2 An example illustrating *RCPFast* algorithm. 42
3.3 An illustration of the optimization strategy based on Theorem 3.3. 48
3.4 Examples of the approximation strategy. 49
3.5 Compression rate tests on synthetic data sets. 54
3.6 Framework comparison on synthetic data sets. 56
3.7 Performance tests with $minpi$ and ϵ on synthetic data sets. . . 58
3.8 Co-location distance computation analysis on synthetic data sets. 59
3.9 Compression rate differences between *RCPMS* and *RCPFast* on synthetic data sets. 60
3.10 Compression rate tests on EPA and POI data sets. 60
3.11 Performance on EPA and POI data sets. 61

4.1 The HESI framework. 73
4.2 A small spatial dataset. 76
4.3 An example of a normal KD-tree *vs.* an SKD-tree with reference to worker locations in Figure 4.2. 80
4.4 Evaluation of tree construction. 96

4.5 Tree operation evaluation. 97
4.6 Overall Performance. 98
4.7 Performance Improvement. 99

Chapter 1

Introduction

Spatial data mining (SDM) is a process of knowledge discovery that the observing data is related to geographical information. Its goal is to identify spatial patterns, identify spatial objects that are potential generators of spatial patterns, or identify the relevant information for explaining the spatial patterns. In this chapter, we give a brief introduction about the characteristic of SDM, address the challenges during the knowledge mining process and investigate the methods of solving these challenges.

Spatial data mining (SDM) becomes an important data mining task due to the explosive growth and pervasive use of spatial data [2]. It aims to discover interesting and potentially useful patterns from large spatial datasets [3]. The primary data object of traditional data mining focuses on transactional data records, with the purpose of identifying customer-buying patterns in market basket data. Several general data mining tools, such as See5/C5.0 and Enterprise Miner, are designed to analyze scientific data, multi-media data, medical data and web data. However, due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation, mining interesting and useful knowledge from spatial datasets is more difficult than extracting the corresponding patterns from traditional numeric and categorical datasets [4].

The input data of SDM can be categorized into two types: non-spatial attribute and spatial attribute. The first attribute refers to those attributes indicating the object's non-spatial features such as name, size, and population of a city. Usually they are processed in the same way as that in classi-

Table 1.1: Relationships among non-spatial and spatial data [1].

Non-spatial relationship	Spatial relationship
Arithmetic	Set-oriented: union, intersection, membership, ...
Ordering	Topological: meet, within, overlap, ...
Is_instance_of	Directional: north, left, above, behind, ...
Subcalss_of	Metric: distance, area, perimeter, ...
Part_of	Dynamic: update, create, destroy, ...
Membership_of	shape-based and visibility

cal data mining. Spatial attribute often includes location information (*e.g.*, longitude, latitude and elevation), and other extended information such as shape and covering area. More specifically, geographical data has the following specific features: (1) various data types, *e.g.*, extended spatial objects (points, lines, or polygons); (2) implicit spatial relationships among the spatial objects, *e.g.*, neighborhood; (3) dependent observations, *e.g.*, spatial or temporal co-located; and (4) spatial autocorrelation among spatial features.

Relationships among non-spatial data are explicit. Examples of relationships are *is_member_of*, *is_subcalss_of*, *is_part_of*, and *ordering*. In contrast, relationships among spatial objects are often implicit, such as *overlap*, *intersect*, *behind* and *is_neighbor_of*. The methods of processing relationships among non-spatial data cannot be easily applied to the spatial scenario. For comparison, Table 1.1 summarizes some examples of relationships among non-spatial and spatial data.

One possible way to deal with implicit spatial relationships is to materialize the relationships into transactional data records and then apply traditional data mining techniques [5]. Though this method is simple and clear, it will cause information loss and produce inaccurate result. Another way to solve this problem is to develop algorithms to incorporate the relationships information into the data mining process [6, 7]. In this way, not only it is able to capture the implicit spatial relationships, but also the efficiency of the mining process can be improved.

The output of SDM often includes spatial patterns that can deliver interesting insight of the original data. Spatial patterns can be divided into four important categories: location predictions, spatial outliers, spatial colocation patterns, and spatial clustering. (1) A predictive model is able to

predict desired events occurring at particular geographic locations. Usually, classification of spatial data in regional economics and natural resources are studied. Two models, the spatial autoregressive model (SAR) [8] and Markov random field (MRF)-based Bayesian classifiers [6] are proposed for effective prediction. (2) A spatial outlier is defined as a spatial object whose non-spatial attribute values differ significantly from those of other spatial objects in its spatial neighborhood. Finding spatial outliers is useful in many geographic information systems [9, 10]. (3) Co-location pattern discovery is a process to identify spatial features (*e.g.*, restaurants, schools) whose instances usually locate in proximity [11]. The approach of mining co-location patterns may look similar to, but, in fact, is quite different from traditional association rule mining problem [5] due to the lack of transactions. (4) Spatial clustering is a process of grouping a set of spatial objects into clusters, so that objects within a cluster are similar with each other according to some similarity measure, but are dissimilar to objects in other clusters [12, 13].

Although existing methods can handle the spatial mining task properly, as the arrival of the big data era, new challenges for SDM are arising.

Firstly, traditional SDM methods usually focus on deterministic datasets, where spatial events occur affirmatively at precise locations. However, it is not always the case in practice. On one hand, the data is inherently uncertain in many applications, especially in sensor environments and moving object applications [14]. On the other hand, artificial noise may be added deliberately for privacy protection [15]. Moreover, considering the continuous nature of spatial data, it is more reasonable to model the location of an instance as a continuous variable (*e.g.*, Gaussian distribution), instead of a discrete one [16]. Hence, data **uncertainty** is ubiquitous in real world and mining patterns from uncertain data has become an interesting and important task in the literature [17, 18, 19, 20, 21].

Secondly, traditional SDM frameworks produce an exponential number of patterns, which makes it hard for users to understand or apply. For example, in terms of spatial co-location pattern mining, it requires a user-specified minimum threshold to find interesting patterns [22]. If the threshold is high, the framework may generate commonsense patterns. How-

ever, with a low threshold, a great number of patterns will be found. Consequently, a huge pattern number will jeopardize the usability of resulted patterns, as it demands great efforts to understand or examine the discovered knowledge. Therefore, it is important to address the **condensity** problem. One key idea of solving this problem is to find an effective way to summarize the resulting patterns, *e.g.*, to find a high-quality representation that describes the complete set of resulted patterns precisely and concisely. Two types of compressed co-location patterns have been explored in the literature: maximal co-location patterns (MCP) [23] and closed co-location patterns (CCP) [24]. However, MCP is a lossy approximation that fails to preserve the prevalence information, and CCP emphasizes the prevalence information too much, which limits the compression power. Therefore, how to summarize co-location patterns effectively and efficiently is a challenge.

Thirdly, spatial data usually involves an individual's location information. However, disclosing individual locations may lead to serious privacy implications. For example, with the leakage of location information, an adversary may invoke a broad spectrum of attacks such as physical stalking, identity theft, and breach of sensitive information including an individual's health status, political and religious views [25]. Therefore, location **privacy** is a critical security issue and it is important to develop secure methods to protect location privacy when learning and using the spatial data. Traditional solutions use randomization perturbation techniques to disturb the location data [26]. However, data perturbation cannot protect data privacy sufficiently, and the randomized data may also deteriorate the mining quality and result in inaccurate models. Therefore, it would be a challenge to protect location privacy with enhanced data security and improved resulting accuracy.

In this thesis, we intend to address these issues by investigating concrete spatial data mining tasks and applications as follows.

To address the uncertainty issue, we study the problem of discovering co-location patterns in the context of continuously distributed uncertain data. In particular, we aim to discover co-location patterns from uncertain spatial data where locations of spatial instances are represented as multivariate Gaussian distributions. We first formulate the problem of Proba-

bilistic Co-location Pattern Mining (PCPM) based on newly defined prevalence measures. When the locations of instances are represented as continuous variables, the major challenges of probabilistic co-location mining lie in the efficient computation of prevalence measures and the verification of the probabilistic neighborhood relationship between instances. We develop an effective probabilistic co-location mining framework integrated with optimization strategies to address the challenges.

To address the condensity issue, we study the problem of Representative Co-location Patterns Mining (RCPM). We first define a covering relationship between two co-location patterns by finding a new measure to appropriately quantify the distance between patterns in terms of their prevalence, based on which the RCPM problem is formally formulated. To solve the problem, we first propose an algorithm called RCPFast, adopting the post-mining framework that is commonly used by existing distance-based pattern summarization techniques. To address the peculiar challenge in spatial data mining, we further propose another algorithm, RCPMS, which employs the mine-and-summarize framework that pushes pattern summarization into the co-location mining process. Optimization strategies are also designed to further improve the performance of RCPMS.

To address the privacy issue, we study the problem of protecting Location Privacy in Spatial Crowdsourcing (LPSC). The objective of spatial crowdsourcing is to outsource a set of spatio-temporal tasks to a set of individual workers who will perform the tasks by physically traveling to the locations of interest. We propose a secure spatial crowdsourcing framework based on encryption, which ensures that all location information will not be released to any party. We solve the challenge of assigning tasks on encrypted data by using homomorphic encryption. Moreover, to overcome the efficiency issue, we propose a novel secure indexing technique with a newly devised SKD-tree to index encrypted worker locations.

The remainder of this thesis is as follows. Chapter 2 investigates the methods of discovering co-location patterns in the context of continuously distributed uncertain data. Chapter 3 discusses the challenges of mining representative co-location patterns and proposes effective solutions. Chapter 4 studies the spatial data application, spatial crowdsourcing, in the privacy-

preserving scenario. Chapter 5 concludes the thesis.

Chapter 2

Mining Co-location Patterns from Uncertain Data

A co-location pattern is a special type of pattern that describes the proximity of spatial features, and is able to provide interesting insights for knowledge discovery. Traditional co-location pattern mining focuses on discovering co-location patterns from deterministic spatial data sets. However, in real world data collected from various sources tend to be uncertain due to measurement noises and errors. It is more appropriate to consider uncertain data. In this chapter, we study the problem in the context of continuously distributed uncertain data. In particular, we aim to discover co-location patterns from uncertain spatial data where locations of spatial instances are represented as multivariate Gaussian distributions. We first formulate the problem of *probabilistic co-location mining* based on newly defined prevalence measures. When the locations of instances are represented as continuous variables, the major challenges of probabilistic co-location mining lie in the efficient computation of prevalence measures and the verification of the probabilistic neighborhood relationship between instances. We develop an effective probabilistic co-location mining framework integrated with optimization strategies to address the challenges. Our experiments on multiple datasets demonstrate the effectiveness of the proposed algorithm.

2.1 Introduction

Co-location mining is an important application in spatial data sets. A co-location pattern is a subset of spatial features whose instances are frequently located close to each other [11]. Spatial co-location patterns yield valuable knowledge for various applications. In Epidemiology, for example, different incidents of diseases may exhibit co-location patterns such that one type of disease tends to occur in spatial proximity of another. In Ecology, different types of animals may behave co-location patterns such as symbiotic relationship and predator-prey relationship [22]. In E-commerce, companies may be interested in discovering types of services (e.g., weather, timetabling and ticketing queries) that are requested by geographically neighboring users, so that they can provide location-sensitive recommendations [27]. Due to its importance, the problem of finding prevalent co-location patterns from spatial data sets has been explored extensively [11, 28, 29, 30, 31, 32, 33].

Traditional co-location pattern mining usually focuses on deterministic data sets, where instances of spatial features occur affirmatively at precise locations. However, it is not always the case in practice. On one hand, the data is inherently uncertain in many applications, especially in sensor environments and moving object applications [34]. On the other hand, artificial noise may be added deliberately for privacy protection [15]. Hence, data uncertainty is ubiquitous in real world and mining patterns from uncertain data has become an interesting and important task in the literature [35].

A few works on mining co-locations from uncertain spatial data have emerged recently, which consider data uncertainty from different aspects. Wang et al. [36] study mining co-location patterns from uncertain spatial data where instances are associated with *existential probabilities*. That is, whether an instance occurs or not is uncertain. However, if it occurs, its location is assumed to be deterministic. In contrast, Liu and Huang [37] explore the problem of co-location mining from uncertain data by recognizing the *location probabilities* of instances. Given an instance, their work considers several (typically 3-5) possible locations within a bounded range and assigns probabilities to indicate how likely the instance occurs at one

of the locations. That is, the location of an instance is modeled as a discrete variable.

Considering the arrival of big data era, coupled with the continuous nature of spatial data, it is very likely that for each instance, a collection of possible locations may be gathered. For example, in the application of interesting constellation discovery in astrophysics, it is common to record the locations of stars in a long time period while the locations may vary every time the stars are observed. In this case, it is more reasonable to model the location of an instance as a continuous variable, instead of a discrete one. Moreover, existing method [37] models the location of an instance as a discrete variable *within a bounded range*, which may cause loss of information. In fact, many practical applications are essentially unbounded, such as RFID positions [16], GSM phone positioning [38] and GPS logs [39]. In these scenarios, the location of an instance is usually represented as a continuous *Gaussian distribution*. Therefore, in this chapter, we focus on the problem of mining co-locations from uncertain spatial data where location of each instance is modeled as a continuous *multivariate Gaussian distribution*, which is widely used in modeling location uncertainty such as in spatial range querying [40] and localization in robotics [41]. To the best of our knowledge, this is the first work that mines co-locations from Gaussian-based uncertain spatial data.

Mining co-location patterns from uncertain spatial data where locations are continuous variables is a challenging problem. Firstly, the existing framework of problem definition cannot be adopted directly because the existing interestingness measures cannot deal with locations modeled as probabilistic distributions. Secondly, the mining process will be computationally expensive and complicated. For example, when locations of instances are represented as probabilistic distributions, expensive integration will be involved to examine whether two instances are probabilistic neighbors.

To address the challenges, we first re-define the interestingness measures to cope with continuously distributed spatial data, based on which the problem of *probabilistic co-location mining* is formulated (Section 2.3). To compute the newly defined prevalence measure, it is essential to find out the probability that an instance supports/participates a feature set. We

propose proper and effective schemes to compute the probability efficiently (Section 2.4). After handling the definition and computation of interestingness measures, a framework for probabilistic co-location mining is put forward (Section 2.5). Observing the bottleneck of the mining process lies in the discovery of probabilistic neighbors of instances, we further devise an optimization strategy to skip verifying the neighborhood relationship between particular instance pairs (Section 2.6).

The main contributions of this chapter are summarized as follows.

- We have formulated the problem of *probabilistic co-location mining* from Gaussian-based uncertain spatial data, based on newly defined prevalence measures.
- We have developed a framework for mining probabilistic co-locations from Gaussian-based spatial data, with effective strategies to address the computation of prevalence and the verification of probabilistic neighborhood relations.
- We have conducted experiments on multiple data sets to examine the effectiveness of the proposed methodologies.

2.2 Related Works

The problem of frequent co-location mining from spatial databases is first introduced by Morimoto [42]. A *support* metric, which is defined as the number of instances of a co-location, is used to measure the prevalence of a co-location pattern. However, the metric does not possess the anti-monotonic property. Shekhar and Huang [11] propose to use *participation ratio* and *minimum participation index* as the interestingness measures that are more statistically meaningful. Various algorithms have been developed to mine prevalent co-location patterns based on the two measures, such as the Apriori-like algorithm [22], the fast algorithm combining the discovery of neighbors with the mining process [27], the join-based algorithm [22], the partial-join algorithm [43], the join-less algorithm [29] and the CPI-tree-based algorithm [44]. Other interestingness measures have also been ex-

plored, such as mining confident co-locations using the *maximum participation ratio* [45] and mining co-locations based on statistic hypothesis [32]. Recently, Qian et al. [46] proposes the *distance variation coefficient* as a new measure to discover regional co-locations.

There are other extensions of co-location mining. Xiong et al. [28] mine co-location patterns from data sets with extended spatial objects such as ling-strings and polygons. Complex co-location pattern mining is studied by Munro and Sun [47] to find negative co-locations. Celik, Kang and Shekhar [31] explore the problem of finding zonal co-locations with dynamic parameters, i.e., repeated specification of zone and interestingness measure values preferred by users. Qian et al. [33] address the problem of mining co-locations without specifying thresholds. Yang et al. studies the high utility co-locoation patterns [48]. There are other studies focusing on real-word applications such as medical cases [49], road networks [50].

Data mining over uncertain data has become an active research area recently [51]. Many research efforts have been devoted to mining frequent patterns from uncertain data [52, 53, 35, 54, 55, 18]. As aforementioned, there are also works on mining co-locations from uncertain data [36, 37]. Our work is different from Wang et. al [36] because we consider location probability instead of existential probability. Our work is different from Liu and Huang [37] since we model location information as continuous variables rather than discrete variables.

2.3 Problem Definitions

In this section, we first review definitions related to co-location mining from deterministic data. Next, we formally define our problem based on redefined prevalence measures in the context of Gaussian-based uncertain spatial data.

2.3.1 Co-location Patterns in Deterministic Data

Given a set of spatial features $\mathcal{F} = \{f_1, f_2, \ldots, f_K\}$, a deterministic spatial data set is a collection of instances/events $\mathcal{E} = \{e_1, e_2, \ldots, e_N\}$. Each $e_i \in \mathcal{E}$

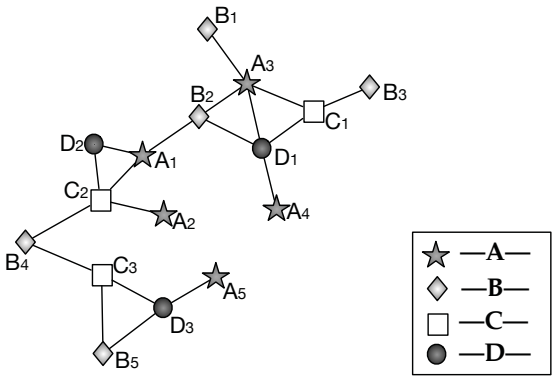


Figure 2.1: An example of deterministic spatial data.

is a vector $\langle event\ identity, spatial\ feature, location \rangle$, indicating its identity, feature type and occurrence location in a spatial framework S. An example of a deterministic spatial data set is given in Figure 2.1, where each symbol represents an event of a specific spatial feature. For simplicity, we use capital letters to denote features, and subscripts to denote event identities. Two events are connected if they belong to different features and if their spatial distance is less than a specified distance threshold τ. Given a deterministic spatial data set, measures related to characterizing the interestingness of a subset of features $F \subseteq \mathcal{F}$ have been defined by Sheckhar and Huang [11].

Definition 2.1. *Given a subset of features $F = \{f_1, f_2, \ldots, f_k\} \subseteq \mathcal{F}$, $E = \{e_1, e_2, \ldots, e_k\} \subseteq \mathcal{E}$ is a **Row Instance (RI)** of F, denoted as $RI(F)$, if $\forall i \in [1, k]$, e_i is an instance of f_i and $\forall i, j \in [1, k], ||e_i - e_j|| \leq \tau$, where $||e_i - e_j||$ refers to the distance between two events.*

For example, in Figure 2.1, $\{A_1, C_2\}$ is a row instance of $\{A, C\}$. Similarly, $\{A_1, C_2, D_2\}$ and $\{A_3, C_1, D_1\}$ are row instances of $\{A, C, D\}$.

Definition 2.2. *The **Table Instance (TI)** of a subset of features $F \subseteq \mathcal{F}$, denoted as $TI(F)$, is the collection of all its row instances $\{RI_1(F), \ldots, RI_m(F)\}$.*

Consider Figure 2.1. $TI(\{A, C\}) = \{\{A_1, C_2\}, \{A_2, C_2\}, \{A_3, C_1\}\}$, and $TI(\{A, C, D\}) = \{\{A_1, C_2, D_2\}, \{A_3, C_1, D_1\}\}$.

Definition 2.3. *Given a subset of features $F = \{f_1, \ldots, f_k\}$, the **Participation Ratio** of a feature $f_i \in F$, denoted as $PR(f_i, F)$, is the fraction of events of feature*

f_i that participate in the table instance of F. That is,

$$PR(f_i, F) = \frac{|\{e_j | e_j \in \widehat{TI}(\{f_i\}), e_j \in \widehat{TI}(F)\}|}{|\{e_j | e_j \in \widehat{TI}(\{f_i\})\}|}, \tag{2.1}$$

where $\widehat{TI}(\cdot)$ is the union of elements in TI set. Hence, the denominator refers to the total number of events of feature f_i and the numerator refers to the distinct number of events of feature f_i that appear in the table instance of F.

For example, consider $TI(\{A, C\}) = \{\{A_1, C_2\}, \{A_2, C_2\}, \{A_3, C_1\}\}$ in Figure 2.1. We have $PR(A, \{A, C\}) = \frac{3}{5}$, since among the five distinct events of feature A, three of them participate in the table instance of $\{A, C\}$. Likewise, we have $PR(C, \{A, C\}) = \frac{2}{3}$.

Definition 2.4. *The **Participation Index** of a subset of features $F = \{f_1, \ldots, f_k\}$, denoted as $PI(F)$, is defined as*

$$PI(F) = \min_{i \in [1,k]} PR(f_i, F). \tag{2.2}$$

For example, consider $\{A, C\}$ in Figure 2.1 again. Since $PR(A, \{A, C\}) = \frac{3}{5}$ and $PR(C, \{A, C\}) = \frac{2}{3}$, then $PI(\{A, C\}) = \min(\frac{3}{5}, \frac{2}{3}) = \frac{3}{5}$.

Definition 2.5. *Given a user-specified threshold min_{PI}, a subset of features $F \subseteq \mathcal{F}$ is a prevalent **Co-location Pattern** if $PI(F) \geq min_{PI}$.*

For instance, suppose the prevalent threshold min_{PI} is 0.5. Then $\{A, C\}$ in Figure 2.1 is a prevalent co-location pattern.

2.3.2 Co-location Patterns in Gaussian-based Data

In this subsection, we model the location of an event as a continuous variable. In particular, given an event e_i, the location of e_i is represented as a d-dimensional Gaussian distribution, $\boldsymbol{e_i} = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(d)})^T$, with its mean location center $\boldsymbol{\mu_i}$ and the corresponding covariance matrix $\boldsymbol{\Sigma_i}$. The probability distribution function is given by

$$P_{\boldsymbol{e_i}}(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma_i}|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_i})^T \boldsymbol{\Sigma_i}^{-1} (\boldsymbol{x} - \boldsymbol{\mu_i})\right], \tag{2.3}$$

where $\boldsymbol{\Sigma_i}^{-1}$ is the inverse matrix and $|\boldsymbol{\Sigma_i}|$ refers to the determinant of the matrix.

In the context of uncertain data, the distance between two events becomes a probabilistic distribution. Therefore, we define the probabilistic neighborhood relationship between a pair of events as follows.

Definition 2.6. *Given a distance threshold τ ($\tau \geq 0$) and a probabilistic neighborhood threshold θ ($0 < \theta < 1$), two events e_i and e_j are **probabilistic neighbors** if the probability that the distance between them is no greater than τ is no less than θ. That is,*

$$\Pr[||\boldsymbol{e_i} - \boldsymbol{e_j}|| \leq \tau] \geq \theta, \tag{2.4}$$

where $||\bullet||$ denotes the distance between two d-dimensional objects. If two events are probabilistic neighbors, we denote $R_{\tau,\theta}(e_i, e_j) = 1$.

Based on the definition of probabilistic neighbors, we can define a *clique instance* of a co-location as follows, corresponding to the concept of row instance in the context of deterministic data.

Definition 2.7. *Given a subset of features $F = \{f_1, \ldots, f_k\}$, a set of events $E = \{e_1, \ldots, e_k\}$ is a **Clique Instance** of F, denoted as $CI(F)$, if $\forall i \in [1, k]$, e_i is an event of f_i, and $\forall i, j \in [1, k]$, $R_{\tau,\theta}(e_i, e_j) = 1$.*

Given a subset of features, we can find a collection of clique instances from the input spatial data. We record the set of clique instances, $\{CI_1(F), CI_2(F), \ldots, CI_m(F)\}$, in a *clique instance table*, denoted as $CIT(F)$.

Recall that in deterministic data, the participation ratio of a feature in a subset of features is computed as the fraction of events of this feature that participate in the collection of row instances of the feature set. However, in the context of uncertain data, whether an event participates in a clique instance is probabilistic. Let $PR(f_i.e_j, F)$ be the probability that the jth event of feature f_i participates in the collection of clique instances of F (we will explain how to compute this value in the next section). Then, the probabilistic participation ratio of a feature in a feature set can be defined as follows.

Definition 2.8. *Given a subset of features $F = \{f_1, \ldots, f_k\}$, the **Probabilistic Participation Ratio** of a feature $f_i \in F$, denoted as $PPR(f_i, F)$, is defined as:*

$$PPR(f_i, F) = \frac{1}{|f_i|} \sum_{j=1}^{|f_i|} PR(f_i.e_j, F), \tag{2.5}$$

where $|f_i|$ refers to the number of events of f_i.

Then, a probabilistic participation index can be defined similarly as in deterministic data.

Definition 2.9. *The **Probabilistic Participation Index** of a subset of features $F = \{f_1, \ldots, f_k\}$ is defined as*

$$PPI(F) = \min_{i \in [1,k]} PPR(f_i, F). \tag{2.6}$$

Based on the newly defined concepts and measures, we formalize the problem of *probabilistic co-location mining* from Gaussian-based uncertain spatial data as follows:

Definition 2.10 (Problem Definition). *Given a set of spatial features $\mathcal{F}$, a set of events $\mathcal{E}$ on $\mathcal{F}$ where each event is associated with a location random variable represented as a d-dimensional Gaussian distribution, a distance threshold τ, a neighborhood probability threshold θ, and a minimal probabilistic participation index threshold min_{PPI}, the objective is to discover the complete set of probabilistic co-location patterns where for each pattern $F \subseteq \mathcal{F}$,*

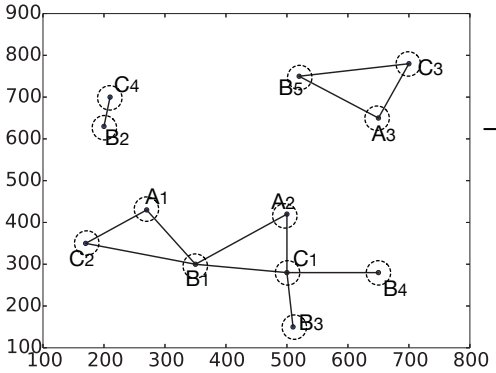
$$PPI(F) \geq min_{PPI}. \tag{2.7}$$

2.4 Probabilistic Participation Ratio Computation

In this section, we discuss how to compute the probabilistic participation ratio of an event of a feature in a feature set, i.e., $PR(f_i.e_j, F)$.

Note that, since the neighborhood relationship between two events is probabilistic, each clique instance of a feature set is also associated with a probability representing how likely the set of events constitutes a clique instance.

Definition 2.11. *Let $CI(F) = \{e_1, \ldots, e_k\}$ be a clique instance of a subset of features $F = \{f_1, \ldots, f_k\}$. Then, we associate a probability with the clique instance*



A	B	C	P[CI(F)]
A_1	B_1	C_2	0.79
A_2	B_1	C_1	0.72
A_3	B_5	C_3	0.85

Figure 2.2: An example of an uncertain spatial data.

as

$$\Pr[CI(F)] = \underbrace{\iint \ldots \int}_{k} \Psi(\boldsymbol{x_1}, \ldots, \boldsymbol{x_k}) \cdot P_{e_1}(\boldsymbol{x_1}) \ldots P_{e_k}(\boldsymbol{x_k}) d\boldsymbol{x_1} \ldots d\boldsymbol{x_k}, \quad (2.8)$$

$$\textit{where } \Psi(\boldsymbol{x_1}, \ldots, \boldsymbol{x_k}) = \begin{cases} 1 & \textit{if } \forall \boldsymbol{x_i}, \boldsymbol{x_j}, ||\boldsymbol{x_i} - \boldsymbol{x_j}|| \leq \tau \\ 0 & \textit{otherwise} \end{cases}$$

Since this numerical integration cannot be calculated analytically, the Monte Carlo method can be adopted with a sufficient number of samplings.

Then, given a collection of uncertain spatial data, each feature set F can be associated with a clique instance table $CIT(F)$ where each clique instance is accompanied with a probability obtained by Eq. (2.8). For example, Figure 2.2 shows a toy example of an uncertain spatial data set, and a clique instance table of the feature set $F = \{A, B, C\}$. The dashed ellipse represents the location of an event is a Gaussian distribution with standard covariance 10. The probability of each clique instance is computed using the example data as specified in Definition 10.

To find the probability of an event of a feature $f_i.e_j$ participates a feature set F, we consider the following two situations. If the event participates in only one clique instance of the feature set (e.g., $CI(F)$), the probability $PR(f_i.e_j, F)$ equals to the existence probability of the clique instance (e.g., $\Pr[CI(F)]$). For example, in Figure 2.2, $PR(A_3, \{A, B, C\}) = 0.85$. If the event participates in more than one clique instance, then we can't simply

add the probabilities of all involved clique instances. For example, consider the event B_1 in Figure 2.2. Since B_1 participates in two clique instances of the feature set $\{A, B, C\}$, adding the probabilities of the two clique instances will result in $PR(B_1, \{A, B, C\}) = (0.79 + 0.72) > 1$, which is incorrect. The reason is that the clique instances of a feature set are not independent. To address the issue, the following lemma gives the correct computation of $PR(f_i.e_j, F)$.

Lemma 2.1. *Let $CIT'(F) = \{CI_1, CI_2, \ldots, CI_m\} \subseteq CIT(F)$ be the set of clique instances of feature set F that an event $f_i.e_j$ participates in. The event participation ratio, $PR(f_i.e_j, F)$, can be given by:*

$$PR(f_i.e_j, F) = \sum_{CI_i \in CIT'} \Pr[CI_i] - \sum_{CI_i, CI_j \in CIT'} \Pr[CI_i, CI_j] + \cdots + (-1)^{m-1} \Pr[CI_1, \ldots, CI_m]. \tag{2.9}$$

The lemma can be proved based on the inclusion-exclusion principle in combinatorial mathematics [56]. As a result, in the example in Figure 2.2, since $\Pr[\{A_1, B_1, C_2\}, \{A_2, B_1, C_1\}] = 0.53$, we have $PR(B_1, \{A, B, C\}) = 0.79 + 0.72 - 0.53 = 0.98$.

Although Lemma 2.1 provides a proper solution to calculate $PR(f_i.e_j, F)$, it suffers the computation efficiency problem, especially when an event participates in a large number of clique instances. That is, when m is large. This is because the number of terms in Eq. (2.9) is proportional to 2^m. Moreover, each $\Pr[CI_i, \ldots, CI_j]$ has to be obtained by sufficient number of samplings, which consumes considerable time.

In fact, since $PR(f_i.e_j, F)$ is the probability that $f_i.e_j$ participates in one of the clique instances in $CIT(F)$, we can skip calculating $\Pr[CI(F)]$ and deal with $PR(f_i.e_j, F)$ directly by using the Monte Carlo method.

Definition 2.12. *Let $CIT'(F) = \{CI_1, CI_2, \ldots, CI_m\} \subseteq CIT(F)$ be the set of clique instances of feature set F that event $f_i.e_j$ participates in. Let $\mathcal{W}$ denote the set of all samples and $CI_i^{(w)}$ represents a certain clique instance exists in the sample*

w. The event participation ratio, $PR(f_i.e_j, F)$, can be given by:

$$PR(f_i.e_j, F) = \frac{\sum_{w \in \mathcal{W}} |CI_1^{(w)} \text{ or } CI_2^{(w)}, \ldots, \text{ or } CI_m^{(w)}|}{|\mathcal{W}|}. \tag{2.10}$$

The Monte Carlo method indicates that, by sampling the data set $\mathcal{W}$ times, the probability $PR(f_i.e_j, F)$ can be obtained as the fraction of data samples where any clique instance involving the event $f_i.e_j$ exists.

2.5 Probabilistic Co-location Mining Framework

In this section we propose the framework for probabilistic co-location pattern mining from Gaussian-based uncertain spatial data.

Before presenting the framework, we first prove that the anti-monotonic property holds for the newly defined measure of probabilistic participation index.

Property 2.1. *(Anti-monotonicity) The probabilistic participation index of a subset of features is monotonically non-increasing with respect to the number of features in the set.*

Proof. Without loss of generality, we consider two feature sets $F_2 = \{f_1, f_2\}$ and $F_3 = \{f_1, f_2, f_3\}$. The objective is to prove $PPI(F_2) \geq PPI(F_3)$. The two probabilistic participation indexes are given by

$$\begin{aligned} PPI(F_2) &= \min(PPR(f_1, F_2), PPR(f_2, F_2)), \\ PPI(F_3) &= \min(PPR(f_1, F_3), PPR(f_2, F_3), PPR(f_3, F_3)) \end{aligned}$$

The key point is then to prove $PPR(f_i, F_2) \geq PPR(f_i, F_3)$. This is because: (1) If $PPR(f_3, F_3)$ is larger than $PPR(f_1, F_3)$ or $PPR(f_2, F_3)$, then it can be ignored due to the min operation and $PPR(f_i, F_2) \geq PPR(f_i, F_3)$ leads to the result. (2) If $PPR(f_3, F_3)$ is smaller than $PPR(f_1, F_3)$ and $PPR(f_2, F_3)$, then if $PPR(f_i, F_2) \geq PPR(f_i, F_3)$ holds, then $PPI(F_3) = PPR(f_3, F_3) < PPR(f_i, F_3) \leq PPR(f_i, F_2)$.

To prove $PPR(f_i, F_2) \geq PPR(f_i, F_3)$, according to Eq. (2.5), we need to prove $PR(f_i.\boldsymbol{e_j}, F_2) \geq PR(f_i.\boldsymbol{e_j}, F_3)$. Let X (resp. X') be a random variable

that indicates whether the event $f_i.e_j \in CIT(F_2)$ (resp. $f_i.e_j \in CIT(F_3)$) occurs. Then $PR(f_i.e_j, F_2) = \Pr[X]$ and $PR(f_i.e_j, F_3) = \Pr[X']$. Since $F_2 \subset F_3$, we have a conclusion that if X' occurs, X must also occur. Hence $\Pr[X] \geq \Pr[X']$. □

Algorithm 2.1 Probabilistic Co-location Mining

Input: A set of events of different features $\mathcal{F}$ with their locations represented as Gaussian distributions, a distance threshold τ, a probabilistic neighborhood threshold θ, a probabilistic participation index threshold min_{PPI}.
Output: A set of probabilistic co-locations $\mathcal{P}$.
Variable: S_k: a set of CITs of size k.
C_k: a set of size k candidate probabilistic co-locations.
P_k: a set of size k probabilistic co-locations.

1: PNT = *gen_probabilistic_neighborhood_table*(τ, θ)
2: $P_1 = \mathcal{F}, k = 2$
3: **while** $(P_{k-1} \neq \emptyset)$ **do**
4: C_k = *gen_candidate_co-locations* (P_{k-1})
5: **if** $k = 2$ **then**
6: $CIT(C_k)$ = *gen_clique_instance_table* (PNT)
7: add $CIT(C_k)$ to S_2
8: **else**
9: $CIT(C_k)$ = *gen_clique_instance_table* (S_{k-1}, PNT)
10: add $CIT(C_k)$ to S_k
11: **for all** $F \in C_k$ **do**
12: $PPI(F)$ = *cal_ppi* $(F, CIT(F))$
13: **if** $PPI(F) \geq min_{PPI}$ **then**
14: add F to P_k
15: $k = k + 1$
16: **return** $\mathcal{P} = P_2 \cup P_3 \cdots \cup P_k$

According to the anti-monotonicity of probabilistic participation index, if a co-location pattern is prevalent, then all its sub-patterns must also be prevalent. Based on this property, an Apriori-like algorithm is developed to discover probabilistic co-location patterns. The main idea is illustrated in Algorithm 2.1. Given a set of Gaussian-based uncertain spatial data, we first construct a probabilistic neighborhood table (PNT) based on Definition 2.6 (line 1). Each entry of PNT is a pair of events that are probabilistic neighbors. Next, we generate size k candidate co-locations from those of

size $k-1$ using the Apriori-join method [22] (line 4). For each candidate co-location pattern, we derive its clique instance table (CIT) correspondingly (lines 5-8). Specifically, if the candidate co-location is of size 2, its CIT can be retrieved directly from PNT. Otherwise, we can construct the CIT of size k from CITs of size $k-1$ and PNT. For example, the clique instance A_1-B_1-C_1-D_1 may be obtained from clique instances A_1-B_1-C_1 and A_1-B_1-D_1, by verifying whether C_1 and D_1 are probabilistic neighbors in PNT. After CITs are generated, we examine whether a candidate is a valid probabilistic co-location pattern by computing the probabilistic participation ratio of each involved feature based on the generated CIT according to Definition 11 (line 12). An iterative loop is then carried out to generate co-locations of size $k+1$ from those of size k.

We observe that one of the major costs of the mining process come from the generation of probabilistic neighborhood table (PNT), which verifies the probabilistic neighbor relationship between a great amount of event pairs. Therefore, in the next section, we address this issue by devising a filtering technique to improve the efficiency of PNT generation.

2.6 Finding Probabilistic Neighbors

It is not an easy task to verify probabilistic neighborhood relationship between events because it involves numerical integration. The Monte Carlo method is usually adopted, which obtains an approximate probability by sufficient number of samplings. However, the sampling progress still engages high computation complexity. We are thus motivated to improve the efficiency by reducing the number of event pairs that need to be compared.

To this end, we propose an efficient filtering technique using Minimum Bounding Sphere (MBS), based on the ρ-region defined in Dong et al. [40].

Definition 2.13. *[40] Consider a Gaussian-based event location variable $\boldsymbol{e_i}$ and the integration of its probability density function $P_{\boldsymbol{e_i}}(\boldsymbol{x})$ over an ellipsoidal region $(\boldsymbol{x}-\boldsymbol{\mu_i})^T\boldsymbol{\Sigma_i^{-1}}(\boldsymbol{x}-\boldsymbol{\mu_i}) \leq r^2$. Let r_ρ be the value of r within which the result of the*

integration equals ρ*,*

$$\int_{(\boldsymbol{x}-\boldsymbol{\mu_i})^T\boldsymbol{\Sigma_i^{-1}}(\boldsymbol{x}-\boldsymbol{\mu_i})\leq r_\rho^2} P_{\boldsymbol{e_i}}(\boldsymbol{x})d\boldsymbol{x} = \rho, \tag{2.11}$$

The ellipsoidal region $(\boldsymbol{x}-\boldsymbol{\mu_i})^T\boldsymbol{\Sigma_i^{-1}}(\boldsymbol{x}-\boldsymbol{\mu_i}) \leq r_\rho^2$ *is called* ρ*-region of* $\boldsymbol{e_i}$*.*

That is, the ρ-region represents an ellipsoidal region in which the probability that an event occurs is ρ. Given a specified probability ρ, the value of r_ρ can be obtained based on the following property:

Property 2.2. *[57] Given the normalized Gaussian distribution*

$$P_{norm}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) = \frac{1}{(2\pi)^{\frac{d}{2}}} \exp\left[-\frac{1}{2}||\boldsymbol{x}||^2\right], \tag{2.12}$$

consider the integration of $P_{norm}(\boldsymbol{x})$ *over* $||\boldsymbol{x}||^2 \leq \widetilde{r}_\rho^2$*. For a given* ρ $(0 < \rho < 1)$*, let* $\widetilde{r}_\rho$ *be the radius within which the integration becomes* ρ*:*

$$\int_{||\boldsymbol{x}||^2\leq \widetilde{r}_\rho^2} P_{norm}(\boldsymbol{x})d\boldsymbol{x} = \rho. \tag{2.13}$$

Then $r_\rho = \widetilde{r_\rho}$ *holds.*

Although this property specifies that we may obtain r_ρ from $\widetilde{r_\rho}$, there is still no way to derive $\widetilde{r_\rho}$ from ρ analytically with Eq. (2.13). Hence, we construct a $(\widetilde{r}_\rho, \rho)$ table in advance. Given a specified ρ, we return the matching $\widetilde{r}_\rho$, or if not matched, return the $\widetilde{r}_\rho$ corresponding to the smallest $\rho^{'}$ that is greater than ρ to guarantee correctness.

2.6.1 Minimum Bounding Sphere

It is difficult to examine the probabilistic neighborhood relationship between two events with locations represented by ellipsoidal ρ-regions. Hence, we adopt the Minimum Bounding Sphere (MBS) that tightly bounds the ρ-region. Examples of MBS of ρ-regions in 2-D space are shown in Figure 2.3. In order to bound the ellipsoid region, the radius of the sphere should be the major axis, which is given by the following property.

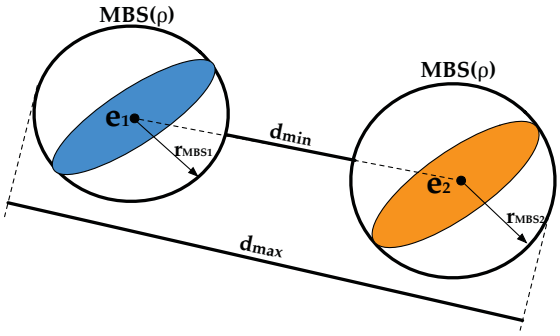


Figure 2.3: Using Minimum Bounding Spheres to bound ρ-regions.

Property 2.3. *Given an ellipsoid* $(\boldsymbol{x}-\boldsymbol{\mu_i})^T{\boldsymbol{\Sigma_i}}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_i}) \leq r_\rho^2$, *the radius* r_{MBS} *of its MBS can be calculated by*

$$r_{MBS} = \frac{\sqrt{r_\rho^2}}{\omega_{min}}, \tag{2.14}$$

where ω_{min} *is the minimum eigenvalue of the covariance matrix* ${\boldsymbol{\Sigma_i}}^{-1}$.

Proof. According to [58], given an ellipsoidal distance $d_A^2(\boldsymbol{p},\boldsymbol{q}) = (\boldsymbol{p}-\boldsymbol{q})^T\boldsymbol{A}(\boldsymbol{p}-\boldsymbol{q})$, a sphere-shape distance function that tightly bounds $d_A^2(\boldsymbol{p},\boldsymbol{q})$ can be obtained by $d_{\text{MBS}}^2(\boldsymbol{p},\boldsymbol{q}) = \omega_{\text{min}}^2(\boldsymbol{p}-\boldsymbol{q})^2$, where ω_{min} is the minimum eigenvalue of matrix $\boldsymbol{A}$. Moreover, it is also proved that given $d_{\text{MBS}}^2(\boldsymbol{p},\boldsymbol{q}) \leq \varepsilon$, the radius of the sphere can be calculated by $\frac{\sqrt{\varepsilon}}{\omega_{\text{min}}}$. If we let $A = {\boldsymbol{\Sigma_i}}^{-1}$ and $\varepsilon = r_{\rho^2}$, then we have $r_{\text{MBS}} = \frac{\sqrt{r_\rho^2}}{\omega_{\text{min}}}$. □

2.6.2 The filtering

We now explain how to quickly verify the probabilistic neighborhood relationship between two events based on MBS. Recall that, if two events e_1 and e_2 are probabilistic neighbors, $\Pr[||\boldsymbol{e_1}-\boldsymbol{e_2}|| \leq \tau] \geq \theta$ holds, where τ and θ are user specified distance threshold and probabilistic neighborhood threshold, respectively. Let *d_min* be the minimum distance between two MBS as illustrated in Figure 2.3. In this case, we have $\Pr[||\boldsymbol{e_1}-\boldsymbol{e_2}|| > d\_min] > \rho^2$, since the probability of an event in the ρ-region is ρ and the probability that it occurs in the corresponding MBS is greater than ρ. That is, $\Pr[||\boldsymbol{e_1}-\boldsymbol{e_2}|| \leq$

$d\_min] \leq 1 - \rho^2$. Let $1 - \rho^2 = \theta$ and consider the critical scenario $d\_min = \tau$, we have $\Pr[||\boldsymbol{e_1} - \boldsymbol{e_2}|| \leq \tau] \leq \theta$. In other words, if $d\_min > \tau$, then the two events are definitely not in a probabilistic neighborhood. We can then filter the pair of events without calculating the numerical integration.

Similarly, we consider the maximum distance between two MBSs. In this case, we have $\Pr[||\boldsymbol{e_1} - \boldsymbol{e_2}|| \leq d\_max] \geq \rho^2$. Let $\rho^2 = \theta$. If $d\_max \leq \tau$, $\Pr[||\boldsymbol{e_1} - \boldsymbol{e_2}|| \leq \tau] \geq \theta$ holds. We can conclude that the two events are probabilistic neighbors straightforwardly.

To sum up, given two events, we apply the following steps to filter event pairs before calculating the numerical integration:

1. Let $\rho = \sqrt{\theta}$ and derive the $d\_max = ||\boldsymbol{\mu_1} - \boldsymbol{\mu_2}|| + r_{\text{MBS1}} + r_{\text{MBS2}}$. If $d\_max \leq \tau$, these two events are directly labeled as probabilistic neighbors.

2. Let $\rho = \sqrt{1 - \theta}$ and derive the $d\_min = ||\boldsymbol{\mu_1} - \boldsymbol{\mu_2}|| - r_{\text{MBS1}} - r_{\text{MBS2}}$. If $d\_min > \tau$, the two events are not probabilistic neighbors.

2.7 Performance Study

We have conducted experiments to evaluate the proposed algorithm using three real data sets. All the experiments are implemented using Python 2.7 on a PC with Intel Core i5 2.5 GHz CPU and 4 GB memory.

2.7.1 Experiment Setup

Three real data sets are used in our experiments. 1) The US National Transportation Atlas Database with Intermodal Terminal Facilities (NTAD-ITF) of 2013 [1], in which every event is a facility with different types such as rail, airport, track, port and inter port. It consists of 5 features and 3087 events in total. 2) The EPA databases [2], which contain environmental activities that affect air, land and water in United States. Different environmental interest types are used as spatial features and each facility represents a spatial

[1] http://www.rita.dot.gov/
[2] http://www.epa.gov/

event. In our experiment, we use the EPA data of Allen Counties in Indiana State, which consists of 23 features and 647 events in total. 3) The points of interest (POI) in California [3] which was used in [59]. There are 63 category types (e.g., dam, school, and bridge) and 104,770 data points.

All the geographic coordinates are transformed to 2-dimensional Cartesian coordinates by Universal Transverse Mercator projection. The uncertainty is generated synthetically by taking the coordinates as the mean values and assigning a covariance matrix to each event. By default, a covariance matrix $\begin{pmatrix} 100^2 & 0 \\ 0 & 100^2 \end{pmatrix}$ is assigned each event. The default sampling times employed by the Monte Carlo method is 1000.

2.7.2 Comparisons with other methods

We first compare our proposed method with other approaches that handle location uncertainties. In particular, we compare our method with two approaches. One is the existing method [37], which considers several possible locations of an event. Given an uncertain spatial data set where locations of events are continuous variables, this method can be applied by randomly sampling several possible location points for each event from its location distribution. The other one simply *determinize* the data by considering the expected locations of events. Then, traditional co-location mining can be applied. We refer to the first method as the *discretization* method, and the second one as the *determinization* method.

2.7.2.1 Comparison with the Determinization Method

As discussed above, one simple and straightforward approach to deal with uncertain data is to remove data uncertainty by considering expected values. Therefore, it would be interesting to investigate the differences between our method that explicitly models the continuous distributions of locations and this simple approach. We conduct experiments to compare the two methods on the EPA data set, by varying the standard covariance σ of location distributions. More precisely, we generate the data uncertainty by using different standard covariances ($\sigma = 10, 100, 300$). We then compare

[3] http://www.usgs.gov/

the number of patterns discovered by the simple *determinization* approach and our method respectively. Other parameters are $\tau = 1050$, $\theta = 0.5$ and $min_{PPI} = 0.4$. Table 2.1 summarizes the experiment results. It can be seen that when $\sigma = 10$, the number of patterns found by both methods are same, so are the particular patterns (shown in the fourth column). As σ increases, the number of patterns output by our method decreases. For example, when σ varies from 10 to 100, the pattern *AirSyntheticMinor-HazardousWasteBiennialReporter* (boldtype in the table) will not be discovered as a valid co-location by our method. When σ varies from 100 to 300, even fewer number of patterns will be found by our method. This is because when σ increases, the location uncertainty of an event becomes more significant, resulting in the lower probability of two events being neighbors. However, the *determinization* method fails to recognize this and outputs the same 8 patterns under different uncertainty degrees.

2.7.2.2 Comparison with the Discretization Method

We further compare our method with the *discretization* approach proposed by Liu and Huang in [37]. In the *discretization* method, each event is associated with several location instances within a region. In our experiment, we model the discrete location uncertainties as follows. For each event, we generate its location instances from its location distribution. The number of location instances per event is decided by a Poisson distribution with mean λ. We run the experiments on the ITF data set. The default parameters are: $\tau = 10000$, $\theta = 0.7$ and $min_{PPI} = 0.3$. By varying λ, we compare the numbers of patterns found by the *discretization* method and our method respectively. The results are shown in Table 2.2.

It can be observed that, when the density λ decreases, fewer number of patterns are discovered by the *discretization* method. This is because when λ gets smaller, fewer location instances will be sampled by the *discretization* method, resulting in the missing of some valid pairs of neighboring events. On the flip side, a large λ invokes more location instances being sampled, so that the result of the discretization method will be similar to that of our method. Our proposed method can thus be regarded as a generalization of the *discretization* method.

Table 2.1: *Determinization* method vs. our method on EPA data.

σ	$\lvert\mathcal{P}\rvert$ by determinization	$\lvert\mathcal{P}\rvert$ by our method	Patterns found by our method
10	8	8	*CESQG-SQG*, *Enforcement-SQG*, *ComplianceActivity-Enforcement*, *CESQG-ComplianceActivity*, ***AirSyntheticMinor-HazardousWasteBiennialReporter***, *CESQG-Enforcement*, *HazardousWasteBiennialReporter-TRIReporter*, *AirSyntheticMinor-CESQG*
100	8	7	*ComplianceActivity-Enforcement*, *CESQG-ComplianceActivity*, *CESQG-Enforcement*, *HazardousWasteBiennialReporter-TRIReporter*, *CESQG-SQG*, *AirSyntheticMinor-CESQG*, *Enforcement-SQG*
300	8	2	*CESQG-ComplianceActivity*, *CESQG-Enforcement*

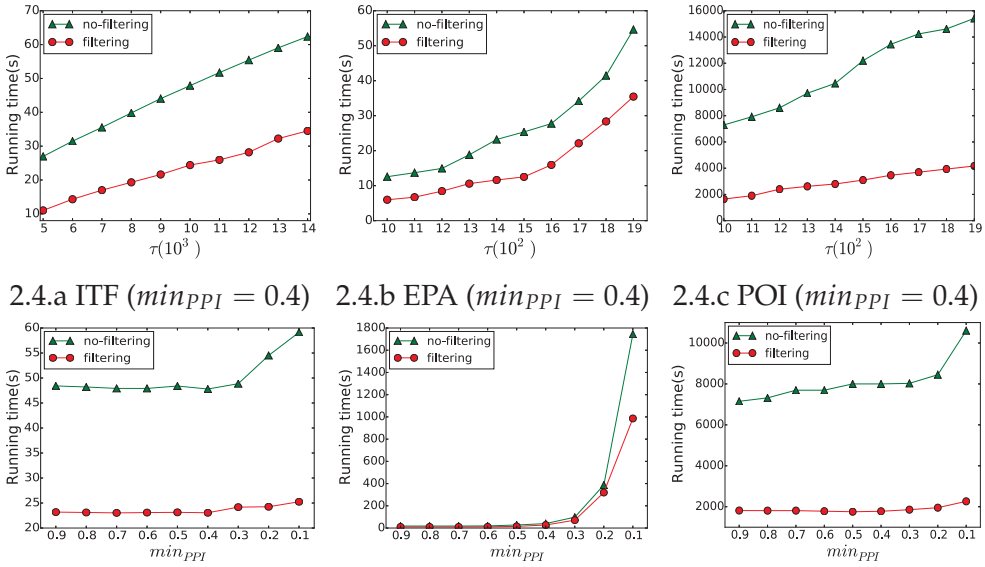
⋆ TRI = Toxics Release Inventory, SQG = Small Quantity Generators and CESQG = Conditionally Exempt Small Quantity Generators

2.7.3 Efficiency of Filtering

Next, we evaluate the efficiency of the filtering method (proposed in Section 6) for finding probabilistic neighbors. We implement our pattern mining framework with and without the filtering technique, and record the running time respectively with respect to the variation of τ and min_{PPI}. The result is shown in Figure 2.4. It can be seen that the filtering technique clearly contributes to the reduction of the running time. Moreover, as the data becomes dense (e.g., the POI data set is denser than the other two), the effect of filtering is more significant.

Table 2.2: *Discretization* method vs. our method on ITF data.

λ	$\lvert\mathcal{P}\rvert$ by our method	$\lvert\mathcal{P}\rvert$ by discritization	Patterns found by discretization
30	3	3	*Airport-Rail, Airport-Truck, Rail-Truck*
10	3	2	*Airport-Rail, Airport-Truck*
3	3	1	*Rail-Truck*



2.4.a ITF ($min_{PPI} = 0.4$) 2.4.b EPA ($min_{PPI} = 0.4$) 2.4.c POI ($min_{PPI} = 0.4$)

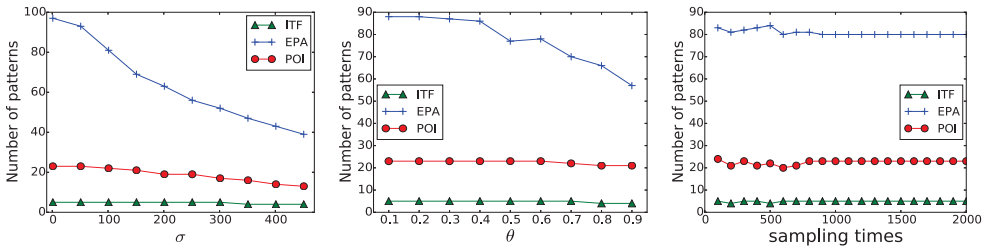
2.4.d ITF ($\tau = 10000$) 2.4.e EPA ($\tau = 1800$) 2.4.f POI ($\tau = 1500$)

Figure 2.4: Evaluation of filtering technique.

2.7.4 Parameter Evaluation

We evaluate our probabilistic co-location mining framework with respect to the variation of other parameters. Figure 2.5 (a) shows the the number of co-location patterns discovered by our method by varying the standard covariance σ. It can be observed that when σ increases, the fewer patterns will be discovered because the data is becoming more uncertain.

Figure 2.5 (b) shows that the number of patterns found by varying the probabilistic neighborhood threshold θ. When θ increases, fewer number of patterns are discovered, especially on the EPA data set due to its larger



2.5.a Evaluating σ 2.5.b Evaluating θ 2.5.c Sampling times

Figure 2.5: Parameter evaluation.

amount of output patterns. The reason is that, when θ gets smaller, fewer event pairs will be valid neighbors.

Since our approach employs the Monte Carlo method to compute the probabilistic prevalence, we also conduct experiments to evaluate the effect of the number of samplings. The results are illustrated in Figure 2.5 (c), from which we observe that for all the three data sets, the number of output patterns becomes stable after sampling 1000 times. That is why 1000 is adopted as the default value in all our experiments.

2.8 Conclusion

This chapter addresses the problem of mining probabilistic co-locations from uncertain spatial data, where locations of events are modeled as Gaussian-based continuous variables. Prevalence measures are redefined to characterize the interestingness in the context of uncertain data, and computed with carefully designed strategy when events participate in multiple clique instances. A proper framework has been developed for probabilistic co-location mining, which integrates an effective filtering strategy to skip expensive verification of probabilistic neighborhood relationship between particular event pairs. Experimental results on multiple datasets demonstrate the efficiency of the proposed algorithm. As extension, several spatial data mining tasks over uncertain data (*e.g.*, k-Nearest Neighbor search problem and frequent trajectory pattern mining) may suffer from similar probability computation problem. The techniques proposed in this chapter may provide interesting insights to other spatial data mining problems.

Chapter 3

Summarizing Spatial Co-location Patterns

A traditional framework of co-location pattern mining produces an exponential number of patterns because of the downward closure property, which makes it hard for users to understand, or apply. To address this condensity issue, in this chapter, we study the problem of mining *representative co-location patterns* (RCP). We first define a covering relationship between two co-location patterns by finding a new measure to appropriately quantify the distance between patterns in terms of their prevalence, based on which the problem of RCP mining is formally formulated. To solve the problem of RCP mining, we first propose an algorithm called *RCPFast*, adopting the *post-mining* framework that is commonly used by existing distance-based pattern summarization techniques. To address the peculiar challenge in spatial data mining, we further propose another algorithm, *RCPMS*, which employs the *mine-and-summarize* framework that pushes pattern summarization into the co-location mining process. Optimization strategies are also designed to further improve the performance of *RCPMS*. Our experimental results on both synthetic and real-world data sets demonstrate that RCP mining effectively summarizes spatial co-location patterns, and *RCPMS* is more efficient than *RCPFast*, especially on dense data sets.

3.1 Introduction

As one of the most fundamental tasks in spatial data mining, *co-location mining* aims to discover co-location patterns where each is a group of spatial features whose instances are frequently located close to each other [11]. Spatial co-location patterns yield important insights for various applications. In epidemiology, for example, different incidents of diseases may exhibit co-location patterns such that one type of disease tends to occur in spatial proximity of another [30]. In ecology, scientists are interested in finding frequent co-occurrences among spatial features, such as drought, substantial increase/drop in vegetation, and extremely high precipitation [60]. In e-commerce, companies may be interested in discovering types of services (e.g., weather, timetabling and ticketing queries) that are requested by geographically neighboring users, so that location-sensitive recommendations can be provided [27]. Due to its importance, the problem of finding prevalent co-location patterns from spatial data has been explored extensively [11, 28, 29, 30, 31, 32, 33].

A common framework of co-location pattern mining uses the frequencies of a set of spatial features participating in a co-location to measure the prevalence (known as *participation index* [11], or PI for short) and requires a user-specified minimum threshold to find interesting patterns. Typically, if the threshold is high, the framework may generate commonsense patterns. However, with a low threshold, a great number of patterns will be found. This is further exacerbated by the downward closure property that holds for the *PI* measure. That is, if a set of features is prevalent with respect to a threshold of *PI*, then all of its subsets will be discovered as prevalent co-location patterns. A huge pattern number will jeopardize the usability of resulted patterns, as it demands great efforts to understand or examine the discovered knowledge.

The key idea of solving this problem is to find an effective way to summarize the co-location patterns, e.g., to find a high-quality representation that describes the complete set of resulted patterns precisely and concisely. Two types of compressed co-location patterns have been explored in the literature: *maximal co-location patterns* (MCP) [61] and *closed co-location patterns*

(CCP) [24]. A co-location pattern is a MCP if it is prevalent itself and none of its super-patterns are prevalent. MCP mining may significantly reduce the number of co-location patterns, but it fails to preserve the prevalence information. It is therefore a lossy approximation. As for the second type, a co-location pattern is a CCP if it is prevalent itself and none of its super-patterns have the same PI as it does. CCP mining not only diminishes the number of co-location patterns but also preserves the complete PI information. However, by emphasizing too much on the PI information, the compression power of CCP mining is limited.

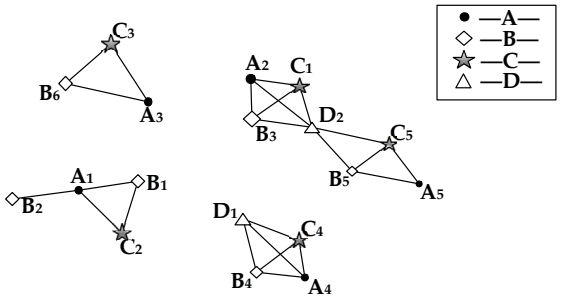


Figure 3.1: A motivating example.

For example, given a spatial data set shown in Figure 3.1, where instances/events of four spatial features, A, B, C and D, are represented by different symbols and edges connecting events denote spatial neighborhood relationships, Table 3.1 lists a set of five prevalent co-location patterns and their corresponding PI in the data set (the definition of PI is provided in Section 2). If MCP mining is adopted, only F_3 will be output as the others are all sub-patterns of F_3. However, F_3 is significantly different from others in terms of their PIs. In contrast, if CCP mining is used, then all patterns will be returned since each of them is a closed pattern. That is, CCP mining provides no compression on this set of patterns. Therefore, to address the limitations of MCP and CCP mining, a method that not only provides optimal compression rate but also preserves reasonable prevalence information will be favored.

Similar idea has been explored in the studies of summarizing frequent itemsets [62, 63, 64]. Xin et al. [62] proposed the notion of a *ϵ-cover relationship* between itemsets. An itemset X_1 is ϵ-covered by another itemset

ID	Feature Sets	Events	PI
F_1	$\{A, B\}$	A_1B_1, A_1B_2, A_2B_3 A_4B_4, A_5B_5, A_3B_6	1
F_2	$\{A, B, C\}$	$A_1B_1C_2, A_2B_3C_1$ $A_3B_6C_3, A_4B_4C_4, A_5B_5C_5$	5/6
F_3	$\{A, B, C, D\}$	$A_2B_3C_1D_2, A_4B_4C_4D_1$	1/3
F_4	$\{B, C, D\}$	$B_3C_1D_2, B_4C_4D_1, B_5C_5D_2$	1/2
F_5	$\{C, D\}$	C_1D_2, C_4D_1, C_5D_2	3/5

Table 3.1: Prevalent patterns in the example.

X_2 if X_1 is a subset of X_2 and $1 - \frac{|T(X_1) \cap T(X_2)|}{|T(X_1) \cup T(X_2)|} \leq \epsilon$, where $T(X_i)$ is the set of supporting transactions of pattern X_i. The goal is then to find a minimum set of representative itemsets that can ϵ-cover all frequent itemsets. In this chapter, we follow their idea and propose to summarize co-location patterns using a set of *representative co-location patterns* (RCPs), which strikes a fine balance between improving compression rate and preserving prevalence information.

However, existing methods for representative itemsets mining cannot be applied directly to representative co-location pattern mining, neither the framework of problem definition nor the mining process. This is mainly because there is no natural notion of *transactions* in co-location mining [11]. Consequently, the original definition of the ϵ-cover relationship cannot be adopted straightforwardly because it is defined on a supporting transaction-based distance measure. Moreover, the mining process will be more complicated as it is more expensive to examine whether a set of feature instances participate in a co-location than checking whether a set of items appear in one transaction.

To formulate the problem of representative co-location pattern mining, we first define a new measure to appropriately quantify the distance between two co-location patterns in terms of their prevalence, based on which the ϵ-cover relationship can be stated on a pair of co-location patterns. To solve the problem of RCP mining, we first propose an algorithm, *RCPFast*, which follows existing distance-based pattern summarization techniques to adopt the *post-mining* framework that finds RCPs from the set of discovered co-location patterns. Observing a peculiar challenge in spatial data mining,

we then develop another algorithm, called *RCPMS*, which employs a *mine-and-summarize* framework to discover RCPs directly from the spatial data. To our knowledge, *RCPMS* is the first work among existing distance-based pattern summarization that pushes summarization into the pattern mining process. Optimization strategies are also devised to further improve the efficiency of *RCPMS*. The main contributions of our research are summarized as follows.

- We formally define the problem of representative co-location pattern mining based on a newly exploited measure to quantify the prevalence proximity between two co-location patterns. To our knowledge, this is the first work that summarizes spatial co-location patterns using distance-based representative patterns.
- We develop two algorithms to discover the set of RCPs, *RCPFast* and *RCPMS*, which adopt fundamentally different mining paradigms and exploit different optimization strategies to improve performance.
- We evaluate the performance of the developed algorithms on both synthetic and real-world data sets. Our experimental results demonstrate the effectiveness of RCP mining, and the efficiency of *RCPMS* compared with *RCPFast*, especially on dense data sets.

The remainder of this chapter is organized as follows. Existing works related to our research are reviewed in Section 3.2. In Section 3.3, we define relevant concepts and formally formulate the problem. Section 3.4 introduces the *RCPFast* algorithm. Section 3.5 describes the *RCPMS* algorithm and optimization strategies. In Section 3.6, we evaluate the performance of the developed algorithms.

3.2 Related Works

The problem of prevalent co-location pattern mining was first introduced by Morimoto [42], where a *support* metric was defined as the number of instances of a co-location and was used to measure the prevalence of a co-location pattern. However, the metric does not possess the anti-monotonic

property. Shekhar and Huang [11] proposed to use *participation ratio* and *minimum participation index* as the interestingness measures that are more statistically meaningful. Various algorithms have been developed to mine prevalent co-location patterns based on the two measures, such as the Apriori-like algorithm [22], the fast algorithm combining the discovery of neighbors with the mining process [27], the join-based algorithm [22], the partial-join algorithm [43] and the join-less algorithm [30]. Other interestingness measures have also been explored, such as mining confident co-locations using the *maximum participation ratio* [45] and mining co-locations based on statistic hypothesis [32].

Frequent pattern summarization has been studied extensively in traditional frequent itemset mining. A variety of concepts have been proposed to find a smaller set of patterns to represent the complete set of frequent patterns, such as maximal patterns [65], closed patterns [66] and non-derivable patterns [67]. One common generalization of closed patterns is the support distance-based approaches [62, 63] which measure the distance between two itemsets based on their supporting transactions and use a pattern to represent/cover its sub-patterns if their distance is no greater than a specified threshold. Although the framework achieves satisfactory compression rate, it cannot not be applied directly to summarize co-location patterns due to the lack of transaction concepts in co-location mining.

Some initial research efforts have been exerted to summarize prevalent co-location patterns. Mining maximal spatial co-location patterns from a large data set was studied in [68]. This work transforms the data into maximal cliques and then considers maximal cliques as transactions for data mining applications. However, the problem of extracting maximal cliques from a graph is known as NP-hard. Wang et al. used an *order-clique-based* approach to identify table instances and mine maximal co-locations [69]. Closed co-location pattern has been studied by Yoo et al. [30]. They presented a framework to mine top-k closed co-location patterns without using minimum prevalence threshold. There are other types of patterns (*e.g.*, condensed spatial co-location patterns [70]) or methods (*e.g.*, ontology-based interactive post-mining method [71]) to address the condensity problem. To our knowledge, our work is the first distance-based approach to sum-

marize co-location patterns using representative patterns, which preserve more prevalence information than maximal co-location patterns and enjoy higher compression rate than closed co-location patterns.

3.3 Preliminary

In this section we first review definitions related to traditional co-location patterns. Then, we introduce a distance metric to measure the prevalence difference between two patterns. Finally, we formally define the problem of representative co-location pattern mining.

3.3.1 Co-location Patterns

Given a set of spatial features $\mathcal{F} = \{f_1, f_2, \ldots, f_K\}$, a spatial data set is a collection of instances/events $\mathcal{E} = \{e_1, e_2, \ldots, e_N\}$, where each $e_i \in \mathcal{E}$ is represented by a vector $\langle$*event_id, spatial_feature_type, location*$\rangle$. We review the measures used to characterize the interestingness of a subset of features $F \subseteq \mathcal{F}$ as follows. Please refer to Section 2.3.1 and [11] for the details.

Definition 3.1. *Given a subset of features $F = \{f_1, \ldots, f_k\} \subseteq \mathcal{F}$, $E = \{e_1, \ldots, e_k\} \subseteq \mathcal{E}$ is a **Row Instance (RI)** of F, denoted as $RI(F)$, if $\forall i \in [1, k]$, e_i is an instance of f_i and $\forall i, j \in [1, k], ||e_i - e_j|| \leq \tau$, where $||e_i - e_j||$ refers to the spatial distance between two events and τ is a user-specified spatial distance threshold.*

Definition 3.2. *Given a spatial data set $\mathcal{E}$ of a set of spatial features $\mathcal{F}$, the **Table Instance (TI)** of a subset of features $F \subseteq \mathcal{F}$, denoted as $TI(F)$, is the collection of all its row instances in $\mathcal{E}$. That is, $TI(F) = \{RI_1(F), \ldots, RI_m(F)\}$.*

For example, consider the spatial data set in Figure 3.1 and $F_5 = \{C, D\}$ in Table 3.1. $\{C_1 D_2\}$ is a RI of F_5. $TI(F_5) = \{C_1\ D_2, C_4 D_1, C_5 D_2\}$.

Definition 3.3. *Given a subset of features $F = \{f_1, \ldots, f_k\}$, the **Participation Ratio** of a feature $f_i \in F$, denoted as $PR(f_i, F)$, is the fraction of events of feature f_i that participate in the table instance of F. That is,*

$$PR(f_i, F) = \frac{|\{e_j | e_j \in TI(\{f_i\}), e_j \in \widehat{TI}(F)\}|}{|TI(\{f_i\})\}|}, \tag{3.1}$$

where $\widehat{TI}(\cdot)$ is the union of elements in TI set. Hence, the denominator refers to the total number of events of feature f_i and the numerator refers to the number of distinct events of feature f_i that appear in the table instance of F.

Definition 3.4. *The **Participation Index** of a subset of features $F = \{f_1, \ldots, f_k\}$, denoted as $PI(F)$, is defined as $PI(F) = \min_{i \in [1,k]} PR(f_i, F)$.*

For example, consider the spatial data set in Figure 3.1 and $F_2 = \{A, B, C\}$ in Table 3.1. Since $PR(A, F_2) = 5/5$, $PR(B, F_2) = 5/6$, $PR(C, F_2) = 5/5$, we have $PI(F_2) = \min(5/5, 5/6, 5/5) = 5/6$.

Definition 3.5. *Given a user-specified threshold $minpi$, a subset of features $F \subseteq \mathcal{F}$ is a **Prevalent Co-location Pattern (PCP)** if $PI(F) \geq minpi$.*

3.3.2 Co-location Distance Measure

A distance measure between traditional frequent itemsets has been proposed in [62]. It compares the supporting transactions of two itemsets and deduces a numerical value as follows, $D(I_1, I_2) = 1 - \frac{|T(I_1) \cap T(I_2)|}{|T(I_1) \cup T(I_2)|}$, where $T(I_i)$ denotes the set of transactions supporting the itemset I_i. However, it is difficult to apply this measure to co-location patterns because there is no natural notion of transactions in co-location mining [11]. One possible solution is to *transactionize* the spatial data to let every maximal clique instance [72] be one transaction. For example, we can derive transactions from the data set in Figure 3.1 as: $t_1 = \{A_3B_6C_3\}$, $t_2 = \{A_1B_2\}$, $t_3 = \{A_1B_1C_2\}$, $t_4 = \{A_2B_3C_1D_2\}$, $t_5 = \{A_5B_5C_5\}$, $t_6 = \{B_5C_5D_2\}$, $t_7 = \{A_4B_4C_4D_1\}$. Then, the supporting transactions of a co-location pattern are the set of the corresponding maximal clique instances. For instance, let $F = \{ABC\}$ be a co-location pattern. $T(F) = \{t_1, t_3, t_4, t_5, t_7\}$. With this manipulation, existing supporting transaction-based distance measure can be applied to co-location patterns directly.

However, one critical problem of this solution is that it needs to find all maximal clique instances first. Maximal clique enumeration is a long-standing problem in graph theory and it is known to be NP-hard. Although many efficient algorithms have been proposed to tackle this problem, such as [68, 73], the complexity is still high when the graph is large and dense.

We thus explore a new distance measure that appropriately quantifies the prevalence difference between two co-location patterns which can be computed efficiently without manipulating the spatial data set.

For simplicity, we denote the set in the numerator of Eq.(3.1) as $E_F(f_i)$ (i.e., $E_F(f_i) = \{e_j | e_j \in TI(\{f_i\}), e_j \in \widehat{TI}(F)\}$). It refers to the set of events of feature f_i that participate in the table instance of F.

Definition 3.6. *Let F_1 and F_2 be two co-location patterns and f be a feature shared by them, namely, $f \in F_1 \cap F_2$, the **Feature Distance** between F_1 and F_2 w.r.t. f is defined as*

$$FD_f(F_1, F_2) = 1 - \frac{|E_{F_1}(f) \cap E_{F_2}(f)|}{|E_{F_1}(f) \cup E_{F_2}(f)|} \tag{3.2}$$

Particularly, if $F_1 \subseteq F_2$, the formula can be rewritten as

$$FD_f(F_1, F_2) = 1 - \frac{|E_{F_2}(f)|}{|E_{F_1}(f)|} \tag{3.3}$$

Definition 3.7. *Given two co-location patterns F_1 and F_2, the **Co-location Distance** between them is defined as*

$$D(F_1, F2) = \begin{cases} \max\limits_{\forall f \in F_1 \cap F_2} FD_f(F_1, F_2), & \text{if } F_1 \cap F_2 \neq \emptyset \\ 1, & \text{otherwise} \end{cases} \tag{3.4}$$

Let us apply this new distance measure to co-location patterns in Table 3.1 to see if it reasonably reflects the distance/proximity between patterns in terms of their prevalence. Firstly, we consider $F_1 = \{A, B\}$ and $F_2 = \{A, B, C\}$. According to the above definitions, $E_{F_1}(A) = E_{F_2}(A) = \{A_1, A_2, \cdots, A_5\}$, $E_{F_1}(B) = \{B_1, B_2, \cdots, B_6\}$, $E_{F_2}(B) = \{B_1, B_3, B_4, B_5, B_6\}$, then $FD_A(F_1, F_2) = 1 - \frac{|E_{F_2}(A)|}{|E_{F_1}(A)|} = 1 - \frac{5}{5} = 0$, $FD_B(F_1, F_2) = 1 - \frac{5}{6} = \frac{1}{6}$. Hence, $D(F_1, F_2) = \max(0, \frac{1}{6}) = \frac{1}{6}$. This small distance value suggests that F_1 and F_2 are quite similar in terms of prevalence. Similarly, let us consider $F_2 = \{A, B, C\}$ and $F_3 = \{A, B, C, D\}$. We can have $D(F_2, F_3) = \max(1 - \frac{2}{5}, 1 - \frac{2}{5}, 1 - \frac{2}{5}) = \frac{3}{5}$, which indicates that the two patterns (F_2, F_3) are quite different. We observe that the new distance measure captures the prevalence distance between co-location patterns appropriately.

3.3.3 Problem Statement

It is natural to consider that we can cluster the collection of co-location patterns using typical cluster methods, e.g., k-means. However, in the RCP problem, the classic cluster methods may not work properly. One of the most crucial problems is that it might select an incorrect RCP for a cluster. This problem can be considered in two aspects: (1) There is no explicit objective to decide a RCP. For instance, suppose F_4 and F_5 in Table 3.1 consist of a cluster, either F_4 or F_5 can be chosen as the RCP according to the distance metric (*i.e.*, $D(F_4, F_5) = 0$). However, it makes no sense to let F_5 represents F_4 because $F_5 \subset F_4$ and it is not able to recover F_4 based on F_5. (2) Even though the co-location distance of two patterns is small, these two patterns may not belong to the same cluster. For example, if $D(\{A, B, C\}, \{B, C, D\})$ is very small, it is not appropriate to let $\{A, B, C\}$ represent $\{B, C, D\}$, or vice versa. To solve this problem, it is reasonable to to define the ϵ-cover relation as follows:

Definition 3.8. *Given two co-location patterns F_1 and F_2, and a real number $\epsilon \in [0, 1]$, we say F_2 ϵ-covers F_1 if*

1. $F_1 \subseteq F_2$;
2. $D(F_1, F_2) \leq \epsilon$.

Then, given a set of prevalent co-location patterns, we can group them into *ϵ-clusters*, where each ϵ-cluster consists of a centroid pattern F_r that ϵ-covers all patterns in the cluster. It seems that we may return centroid patterns of ϵ-clusters as representative patterns. However, by doing so, we restrict the representative patterns to be prevalent themselves (i.e. $PI(F_r) \geq minpi$). The minimum number of representative co-location patterns that can be achieved using this method is the number of MCPs.

In [62], it shows that an itemset only needs to satisfy a relaxed condition (i.e., $Supp(X) \geq (1 - \epsilon) * minsupp$) to ϵ-cover a frequent itemset. We find that this property holds as well for our newly defined distance measure and the induced ϵ-cover relationship.

Let us consider two co-location patterns F_1 and F_2, where $F_1 \subset F_2$ and F_1

is prevalent, $PI(F_1) \geq minpi$. According to the definition of PI, we have

$$PI(F_1) = \min_{\forall f \in F_1} \frac{|E_{F_1}(f)|}{|TI(\{f\})|} \geq minpi \tag{3.5}$$

If F_2 is able to ϵ-cover F_1, then $D(F_1, F_2) \leq \epsilon$. That is,

$$\max_{\forall f \in F_1} (1 - \frac{|E_{F_2}(f)|}{|E_{F_1}(f)|}) \leq \epsilon \tag{3.6}$$

From the above two equations, we have

$$\begin{aligned} \forall f \in F_1, |E_{F_2}(f)| &\geq (1-\epsilon) * |E_{F_1}(f)| \\ &\geq (1-\epsilon) * minpi * |TI(\{f\})| \end{aligned} \tag{3.7}$$

Hence,

$$\forall f \in F_1, \frac{|E_{F_2}(f)|}{|TI(\{f\})|} \geq (1-\epsilon) * minpi \tag{3.8}$$

Recall that, the PI of F_2 can be computed as follows,

$$PI(F_2) = \min_{\forall f \in F_1, f' \in F_2 \setminus F_1} (\frac{|E_{F_2}(f)|}{|TI(\{f\})|}, \frac{|E_{F_2}(f')|}{|TI(\{f'\})|}) \tag{3.9}$$

Thus, as long as we require $PI(F_2) \geq (1-\epsilon) * minpi$, the condition in Eq. (3.8) can be satisfied, no matter $\frac{|E_{F_2}(f)|}{|TI(\{f\})|}$ is greater than $\frac{|E_{F_2}(f')|}{|TI(\{f'\})|}$ or the other way around. That is, to ϵ-cover a prevalent co-location pattern F_1, F_2 only needs to be prevalent with respect to a lower threshold $minpi^* = (1-\epsilon) * minpi$. Our experimental results in Section 3.6 show that this relaxation contributes to an improved compression rate.

Definition 3.9 (Problem Statement). *Given a set of spatial features $\mathcal{F}$, a spatial data set $\mathcal{E}$ on $\mathcal{F}$, a spatial distance threshold τ, a co-location distance threshold ϵ, and a prevalence threshold $minpi$, the problem of representative co-location pattern (RCP) mining is to discover a minimal set of co-location patterns $\mathcal{R}$ such that: (1) For all $F_r \in \mathcal{R}, PI(F_r) \geq (1-\epsilon) * minpi$; (2) For any prevalent co-location patterns F, i.e., $PI(F) \geq minpi$, there exits a $F_r \in \mathcal{R}$ s.t. F_r ϵ-covers F.*

3.4 The *RCPFast* Algorithm

In this section, we first introduce an algorithm, *RCPFast*, which follows existing distance-based pattern summarization approaches to mine RCPs by adopting a *post-mining* framework.

Similar to [62], the mining framework of *RCPFast* consists of three stages. Stage 1 discovers two sets of prevalent co-location patterns, PCP and PCP^*, with respect to $minpi$ and $(1-\epsilon)*minpi$, respectively. The objective is then to select minimal number of patterns from PCP^* to cover all patterns in PCP.

Stage 2 generates the complete coverage information by finding all prevalent co-location patterns $F \in PCP$ that can be ϵ-covered by each pattern $F_r \in PCP^*$. All prevalent co-location patterns ϵ-covered by F_r is stored in $set(F_r)$.

Stage 3 finds the set of desired RCPs based on the coverage information. As discussed in [62], this is a set cover problem which is NP-hard. It can be solved by a greedy strategy that always selects the representative pattern that covers the most number of prevalent co-location patterns. According to [74], the relation between the number of RCPs selected by the greedy solution and the number of the optimal ones is bounded by Theorem 3.1.

Theorem 3.1. *Given a set of prevalent co-location patterns PCP w.r.t. $minpi$, a set of prevalent co-location patterns PCP^* w.r.t. $(1-\epsilon)*minpi$, let the number of RCPs generated using the greedy set cover algorithm be $\mathcal{C}_g$, and the number of optimal RCPs be $\mathcal{C}^*$, then $|\mathcal{C}_g| \leq |\mathcal{C}^*| \times H(\max_{F_r \in PCP^*} |set(F_r)|)$, where $H(n) = \sum_{k=1}^{n} \frac{1}{k}$.*

The time complexity of the greedy algorithm is $O(\sum_{F_r \in PCP^*} |set(F_r)|)$. Thus the computational cost of *RCPFast* mainly comes from the first two stages.

For the first stage, mining prevalent co-location patterns is a well-studied topic. Many efficient algorithms have been proposed, such as the spatial-join method [22] and the join-less method [30]. Note that it is unnecessary to run the mining process twice to discover the two sets of PCP and PCP^*. We can find prevalent patterns w.r.t. $(1-\epsilon)*minpi$ first, and then filter the results to obtain those prevalent w.r.t. $minpi$.

For the second stage, the bottleneck lies in the computations of co-location distance between two patterns to verify the ϵ-cover relationship. The complexity of generating the complete coverage information is $O(|PCP| * |PCP^*|)$, which will become a performance issue when there are many prevalent patterns. Therefore, we aim to exploit strategies to skip verifying the ϵ-cover relationship for as many pairs of patterns as possible.

Theorem 3.2. *Given three co-location patterns F_1, F_2, and F_3 s.t. $F_1 \subseteq F_2 \subseteq F_3$, if F_3 ϵ-covers F_1, then F_2 ϵ-covers F_1.*

Proof. From $D(F_1, F_3) \leq \epsilon$, we have $\forall f \in F_1$,

$$1 - \frac{|E_{F_3}(f)|}{|E_{F_1}(f)|} \leq \epsilon.$$

Because $\forall f \in F_1$, $|E_{F_2}(f)| \geq |E_{F_3}(f)|$. Thus, we have

$$1 - \frac{|E_{F_2}(f)|}{|E_{F_1}(f)|} \leq 1 - \frac{|E_{F_3}(f)|}{|E_{F_1}(f)|} \leq \epsilon,$$

That is, $D(F_1, F_2) \leq \epsilon$, which proves the result. □

According to the theorem, we are allowed to skip computing co-location distance for certain pairs of co-location patterns. For example, as shown in Figure 3.2 (a), if we have found that $\{A, B, C, D\}$ ϵ-covers $\{A, B\}$, then we conclude immediately that $\{A, B, C\}$ ϵ-covers $\{A, B\}$ and $\{A, B, D\}$ ϵ-covers $\{A, B\}$ without computing their corresponding co-location distances. To maximize the benefit introduced by Theorem 3.2, we order the co-location patterns according to pattern lengths. Then, the procedure of *RCPFast* is illustrated in Algorithm 3.1.

Algorithm 3.1 follows the three-stage framework. The first stage (lines 1-2) mines two sets of prevalent co-location patterns and the third stage (lines 15-19) discovers the RCPs using a greedy strategy. The second stage starts with sorting the patterns in PCP^* in decreasing order of pattern length, and sorting patterns in PCP in the reverse order (line 3). Then, a candidate list (*CandList*) is constructed for each representative pattern F_r in PCP^*, which stores all prevalent patterns that may be ϵ-covered by F_r (lines 4-7). Lines 8-14 find the complete coverage information for each pattern F_r in PCP^* by

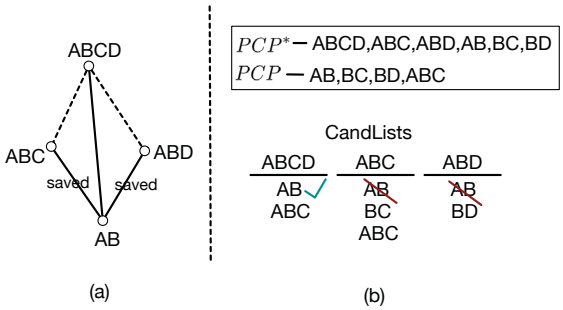


Figure 3.2: An example illustrating *RCPFast* algorithm.

implementing the optimization enabled by Theorem 3.2. In particular, once it is confirmed that F_r ϵ-covers a prevalent pattern F (line 10), we find a set of patterns $\mathcal{Q} \subseteq PCP^*$ where each $Q \in \mathcal{Q}$ is a sub-pattern of the current F_r and a super-pattern of F (line 12). According to Theorem 3.2, F can be immediately added to $set(Q)$ (line 14).

Note that, the purpose of sorting PCP and PCP^* in the specified orders is to allow early discovery of ϵ-cover relationship between a representative pattern and its short sub-patterns so that more pairs of patterns can be skipped for co-location distance computation. For example, Figure 3.2 (b) shows the candidate lists of three representative patterns, $ABCD$, ABC and ABD. Due to the ordering of patterns, we examine first whether $ABCD$ ϵ-covers AB. If it happens, we can delete AB from $CandList(ABC)$ and $CandList(ABD)$ because AB should be covered by these two patterns according to Theorem 3.2. Therefore, the computations of $D(ABC, AB)$ and $D(ABD, AB)$ are omitted.

3.5 The *RCPMS* Algorithm

Recall that, to verify the ϵ-cover relationship in the second stage of *RCPFast*, we need to compute the co-location distance between two patterns, which requires the *table instance* information of the corresponding patterns. However, the output of prevalent co-location pattern mining in the first stage contains only the prevalent patterns as well as their PI information. It may not be an issue for frequent itemset summarization as the supporting

Algorithm 3.1 *RCPFast*

Input: (1) A set of spatial events $\mathcal{E}$, (2) a spatial distance threshold τ, (3) a prevalence threshold $minpi$,(4) a co-location distance threshold ϵ.
Output: The set of RCPs $\mathcal{R}$

```
1: PCP = MinePCP(E, τ, minpi)
2: PCP* = MinePCP(E, τ, (1 − ε) ∗ minpi)
3: Sort PCP* in decreasing order of pattern length, and PCP in increasing
   order of pattern length.
4: for all F_r ∈ PCP* do
5:    for all F ∈ PCP do
6:       if F ⊆ F_r then
7:          Insert F into CandList(F_r)
8: for all F_r ∈ PCP* do
9:    for all F ∈ CandList(F_r) do
10:      if F_r ε-covers F then
11:         Insert F into set(F_r)
12:         Find a set of patterns Q ⊆ PCP* s.t. ∀Q ∈ Q, F ⊆ Q ⊆ F_r
13:         for all Q ∈ Q do
14:            Remove F from CandList(Q) to set(Q)
15: while PCP ≠ ∅ do
16:   Find a F_r that maximizes |set(R)|
17:   for all F ∈ set(F_r) do
18:      Delete F from PCP
19:   R = R ∪ {F_r}
20: Return R
```

transactions of an itemset can be retrieved easily. However, for spatial data mining, it is expensive to re-scan the data to obtain the table instance of a co-location pattern whenever it is required. One possible solution is to output the information of table instances as additional results. However, if the information is stored in disk, extra I/O cost will be incurred. If the information is stored in memory, it will become problematic when the number of patterns is huge. Therefore, we are motivated to push coverage validation into the co-location mining process, thereby integrating the first and the second stages in order to address the table instance acquisition problem.

Based on the idea, we devise the *RCPMS* algorithm that employs a novel *mine-and-summarize* framework, while all existing distance-based pattern summarization techniques adopt the *post-mining* paradigm. More specifi-

Algorithm 3.2 *RCPMS*

Input: Same as *RCPFast*.
Output: Same as *RCPFast*.

```
1: P_1 = F, k = 2
2: while P_{k-1} ≠ ∅ do
3:    C_k = gen_candidate_colo(P_{k-1})
4:    for all C ∈ C_k do
5:       pi = calculate_PI(C)
6:       if pi ≥ (1 − ε) ∗ minpi then
7:          D_Table ← cal_preval_child_dis(C)
8:          set(C) = gen_cover_set(C, C, 0)
9:          if pi ≥ minpi then
10:            Insert C into P_k and set(C)
11:   k = k + 1
12: Obtain RCPs using the greedy algorithm
```

cally, whenever a representative pattern, prevalent w.r.t. $(1-\epsilon) * minpi$, is discovered, all prevalent patterns, w.r.t. $minpi$, which can be ϵ-covered by it will be found. The feasibility of this idea is supported by the following two facts.

1. Traditional prevalent co-location pattern mining algorithms usually use an Apriori-based level-wise scheme to generate patterns [22, 30]. When a representative pattern is mined, all its prevalent sub-patterns have already been found. Hence, it is sufficient to find the coverage information for the current representative pattern.

2. When a representative pattern is output in the mining process, its information of table instance is available, which can be used to compute its co-location distances with its sub-patterns. For its sub-patterns, we store their table instance information in memory if they are child or immediate sub-patterns of the current representative pattern (e.g., $F \subset F_r$ and $|F_r| - |F| = 1$). Otherwise, we will retrieve the table instance information of a sub-pattern F (e.g. $F \subset F_r$ and $|F_r| - |F| > 1$) only if the ϵ-cover relationship between F_r and F cannot be inferred using our devised optimization and approximation strategies, which will be discussed in subsections 4.1 and 4.2.

The general idea of *RCPMS* is summarized in Algorithm 3.2. In the be-

ginning, it assigns all unique spatial features to P_1 (line 1). From line 2 to line 11, an iterative process is used to generate patterns of length k from patterns of length $k-1$. In particular, line 3 calls the function *gen_candidate_colo* to generate candidate co-location patterns (e.g., using an Apriori-like strategy). For each candidate co-location pattern, we first calculate its PI (line 5). If the candidate pattern is prevalent w.r.t. $(1-\epsilon) * minpi$ (line 6), we compute its co-location distances with its child sub-patterns which are prevalent w.r.t. $minpi$ and store it in a distance table (line 7). Note that, only the co-location distances between the current pattern and its prevalent child sub-patterns need to be computed at this stage. As discussed later, its co-location distances with other descendent prevalent sub-patterns will be computed only if they can't be inferred using our proposed optimization and approximation strategies. In line 8, we call the method *gen_cover_set* to find all prevalent sub-patterns that can be covered by the current representative pattern. Finally, if the current pattern is prevalent w.r.t. $minpi$, it should be used to generate candidate patterns in the next round and should be included into its own cover set (lines 9 and 10). Line 12 is the same as the third stage of *RCPFast* which finds the minimal RCPs.

In the following, we describe the details of the function *gen_cover_set* which finds all prevalent sub-patterns that can be ϵ-covered by the current representative pattern. Before presenting the function, we first introduce an optimization strategy and an approximation strategy that are used by the function.

3.5.1 Optimization Strategy

Note that, the optimization strategy used by *RCPFast* (i.e., *Theorem* 3.2) is not applicable here. This is because when we output a representative pattern of length k in *RCPMS*, the coverage information of representative patterns of length $(k-1)$ has already been found. Therefore, we exploit a new optimization strategy based on the following theorem:

Theorem 3.3. *Given three co-location patterns F_1, F_2 and F_3 such that $F_1 \subseteq F_2 \subseteq$*

F_3, it holds that

$$D(F_1, F_2) + D(F_2, F_3) \geq D(F_1, F_3). \tag{3.10}$$

Before we prove this theorem we introduce an auxiliary lemma.

Lemma 3.1. *Let $S_i = (n_{ia}, n_{ib}, n_{ic})$, $1 \leq i \leq m$ be tuples such that each contains three non-negative integers which satisfy $n_{ia} \geq n_{ib} \geq n_{ic}$, then the following inequality holds:*

$$\max_{1 \leq i \leq m}(1 - \frac{n_{ib}}{n_{ia}}) + \max_{1 \leq i \leq m}(1 - \frac{n_{ic}}{n_{ib}}) \geq \max_{1 \leq i \leq m}(1 - \frac{n_{ic}}{n_{ia}}) \tag{3.11}$$

Proof. For simplicity, let $D_1(i) = (1 - \frac{n_{ib}}{n_{ia}})$, $D_2(i) = (1 - \frac{n_{ic}}{n_{ib}})$, $D_3(i) = (1 - \frac{n_{ic}}{n_{ia}})$. The proof can be completed in two stages.

1. Prove $\forall i$, $D_1(i) + D_2(i) \geq D_3(i)$. This can be done by verifying

$$D_1(i) + D_2(i) - D_3(i) = \frac{(n_{ia} - n_{ib})(n_{ib} - n_{ic})}{n_{ia} n_{ib}} \geq 0$$

2. Suppose $\exists\ \tilde{l_1}$, $\tilde{l_2}$, $\tilde{l_3}$ such that $D_1(\tilde{l_1}) = \max_{1 \leq i \leq m} D_1(i)$, $D_2(\tilde{l_2}) = \max_{1 \leq i \leq m} D_2(i)$, $D_3(\tilde{l_3}) = \max_{1 \leq i \leq m} D_3(i)$, because $D_1(\tilde{l_1}) \geq D_1(\tilde{l_3})$ and $D_2(\tilde{l_2}) \geq D_2(\tilde{l_3})$, we have

$$D_1(\tilde{l_1}) + D_2(\tilde{l_2}) - D_3(\tilde{l_3}) \geq D_1(\tilde{l_3}) + D_2(\tilde{l_3}) - D_3(\tilde{l_3}) \geq 0.$$

To sum up, the inequality (3.11) holds. □

Now we provide the proof of Theorem 3.3.

Proof. First of all, because $F_1 \cap F_2 = F_1$, $F_1 \cap F_3 = F_1$, and $F_2 \cap F_3 = F_2$, according to the definitions of co-location distance, we have

$$D(F_1, F_2) = \max_{\forall f \in F_1}(1 - \frac{|E_{F_2}(f)|}{|E_{F_1}(f)|}),$$

$$D(F_2, F_3) = \max_{\forall f \in F_1, f' \in F_2 \setminus F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_2}(f)|}, 1 - \frac{|E_{F_3}(f')|}{|E_{F_2}(f')|}),$$

$$D(F_1, F_3) = \max_{\forall f \in F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_1}(f)|}).$$

Note that we divide $D(F_2, F_3)$ into two parts, such that the former only involves the features in F_1, and the latter involves the remaining. The value of $D(F_2, F_3)$ depends on the relations between the two parts. It can be divided into two situations.

1. $D(F_2, F_3)$ is only related to the former part, or

$$\max_{\forall f \in F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_2}(f)|}) > \max_{\forall f' \in F_2 \setminus F_1}(1 - \frac{|E_{F_3}(f')|}{|E_{F_2}(f')|}).$$

   In this case, $D(F_2, F_3) = \max_{\forall f \in F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_2}(f)|})$. Because $\forall f \in F_1$, $E_{F_3}(f) \subseteq E_{F_2}(f) \subseteq E_{F_1}(f)$ holds, we have $|E_{F_1}(f)| \geq |E_{F_2}(f)| \geq |E_{F_3}(f)|$. Let each $f \in F_1$ binds to a tuple in Lemma 3.1 by building the corresponding relations: $n_{ia} = |E_{F_1}(f)|, n_{ib} = |E_{F_2}(f)|, n_{ic} = |E_{F_3}(f)|$. Therefore by lemma 3.1, we have

$$\max_{\forall f \in F_1}(1 - \frac{|E_{F_2}(f)|}{|E_{F_1}(f)|}) + \max_{\forall f \in F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_2}(f)|}) \geq \max_{\forall f \in F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_1}(f)|}),$$

   which means $D(F_1, F_2) + D(F_2, F_3) \geq D(F_1, F_3)$.

2. $D(F_2, F_3)$ only depends on the latter part, that is

$$D(F_2, F_3) = \max_{\forall f' \in F_2 \setminus F_1}(1 - \frac{|E_{F_3}(f')|}{|E_{F_2}(f')|}),$$
$$\max_{\forall f' \in F_2 \setminus F_1}(1 - \frac{|E_{F_3}(f')|}{|E_{F_2}(f')|}) \geq \max_{\forall f \in F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_2}(f)|}).$$

   According to the first situation, we have

$$\begin{aligned}
&D(F_1, F_2) + D(F_2, F_3) - D(F_1, F_3) \\
&= \max_{\forall f \in F_1}(1 - \frac{|E_{F_2}(f)|}{|E_{F_1}(f)|}) + \max_{\forall f' \in F_2 \setminus F_1}(1 - \frac{|E_{F_3}(f')|}{|E_{F_2}(f')|}) - \max_{\forall f \in F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_1}(f)|}) \\
&\geq \max_{\forall f \in F_1}(1 - \frac{|E_{F_2}(f)|}{|E_{F_1}(f)|}) + \max_{\forall f \in F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_2}(f)|}) - \max_{\forall f \in F_1}(1 - \frac{|E_{F_3}(f)|}{|E_{F_1}(f)|}) \geq 0.
\end{aligned}$$

To sum up, the inequality (3.10) holds, which proves the result. □

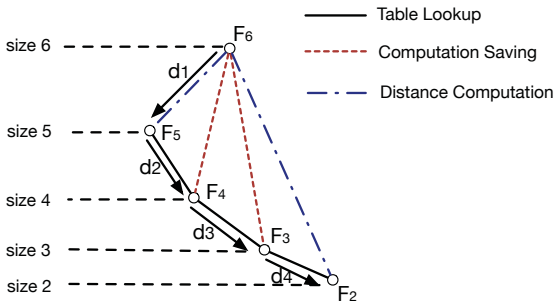


Figure 3.3: An illustration of the optimization strategy based on Theorem 3.3.

Figure 3.3 illustrates how to use Theorem 3.3 to skip computing co-location distance between a representative pattern and its non-child prevalent sub-patterns. Each F_i is a co-location pattern of size i. Different types of lines represent different ways of obtaining the co-location distance. Suppose $d_1 + d_2 + d_3 \leq \epsilon$ and $d_1 + d_2 + d_3 + d_4 > \epsilon$. Suppose F_6 is the current representative pattern and we have computed its co-location distance with its child sub-pattern F_5, *i.e.*, $D(F_6, F_5) = d_1$, stored in the *D_Table* (e.g., line 7 in Algorithm 3.2). Next, we need to examine whether F_6 ϵ-covers F_5's child, e.g., F_4. Note that $D(F_5, F_4) = d_2$ should have been computed and stored in the *D_Table* when outputting F_5 in the previous round. According to Theorem 3.3, we infer that $D(F_6, F_4) < D(F_6, F_5) + D(F_5, F_4) = d_1 + d_2$. Therefore, if $d_1 + d_2 \leq \epsilon$, we can conclude that F_6 ϵ-covers F_4 without computing $D(F_6, F_4)$. Similarly, when examining whether F_6 ϵ-covers F_3, which is a child sub-pattern of F_4, we have $D(F_6, F_3) < D(F_6, F_5) + D(F_5, F_4) + D(F_4, F_3) = d_1 + d_2 + d_3$. As indicated in the figure, $d_1 + d_2 + d_3 \leq \epsilon$, we conclude that F_6 ϵ-covers F_3 and skip computing the distance $D(F_6, F_3)$. When it comes to F_2, since $d_1 + d_2 + d_3 + d_4 > \epsilon$, we have to compute the exact value of $D(F_6, F_2)$ (we will have to re-gain the table instance of F_2 in this case). Therefore, in this particular example, Theorem 3.3 enables us to skip two of the three co-location distance computations (i.e., $D(F_6, F_4)$, $D(F_6, F_3)$ and $D(F_6, F_2)$).

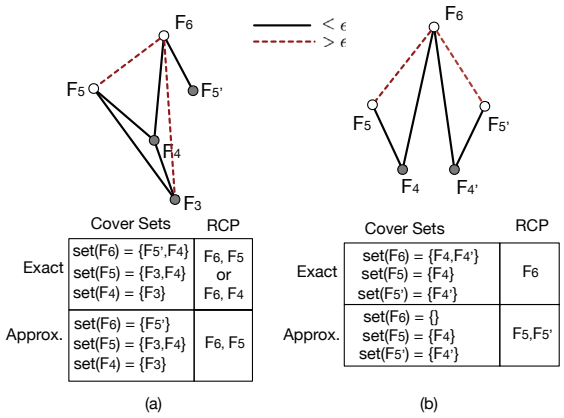


Figure 3.4: Examples of the approximation strategy.

3.5.2 Approximation Strategy

Although Theorem 3.3 can reduce distance computation for a certain number of pairs of patterns, the effectiveness of this single strategy may not be sufficient. Therefore, we further exploit an approximation strategy which substantially improves the computation efficiency by slightly sacrificing the compression rate.

Recall that, in Figure 3.3, only if the co-location distance between the current representative pattern (e.g., F_6) and its child prevalent sub-pattern (e.g., F_5) is smaller than ϵ, we may use the optimization strategy to infer the distance between F_6 and F_4 (F_3). Otherwise, we have to compute the distance between F_6 and F_4 (F_3), which is expensive since we have to re-gain the table instance of F_4 (F_3). Therefore, we consider the following approximation strategy.

If a representative pattern F_r cannot ϵ-cover its prevalent child sub-pattern F, we skip considering whether F_r ϵ-covers any descendant sub-pattern of F.

For example, in Figure 3.3, if the co-location distance between F_6 and F_5 is greater than ϵ, all F_4, F_3 and F_2 will not be included in $set(F_6)$.

Figure 3.4 provides two examples to illustrate the influence of the approximation strategy. In Figure 3.4 (a), let's assume the set of PCP that need to be summarized are F_5', F_4 and F_3, where F_3 is a child sub-pattern of F_4, which is a child sub-pattern of F_5. F_5' is a sibling pattern of F_5 (e.g., ABC and ABD). The exact coverage information shows that F_6 ϵ-covers F_4. How-

ever, since F_6 does not ϵ-cover F_5, F_4 is removed from $set(F_6)$ according to the approximation strategy. If using the greedy algorithm to find RCPs, the final number of RCPs found from the exact cover sets will be 2, which is the same as the final number of RCPs found from the approximate cover sets. It indicates that the approximation strategy does not incur any difference to the final number of RCPs under this situation.

In contrast, Figure 3.4 (b) shows an example where this approximation strategy will result in difference in the final number of RCPs. In this example, suppose the set of PCP are F_4 and F_4'. The complete coverage information shows that F_6 ϵ-covers both F_4 and F_4'. Since F_6 does not ϵ-cover F_5 or F_5', F_4 and F_4' are removed from $set(F_6)$ in the approximate cover sets. Consequently, the final number of RCPs found from the exact cover sets is 1 while the final number found from the approximate cover sets is 2.

In general, we have the following lemma, implying that the final number of RCPs generated from the incomplete cover sets, produced by the approximation strategy, will be no smaller than the final number of RCPs generated from the complete cover sets.

Lemma 3.2. *Let $\mathcal{P}$ be a set of representative patterns with non-empty cover sets. That is, $\mathcal{P} \subseteq PCP^*$ and $\forall P \in \mathcal{P}$, $|set(P)| > 0$. Let $\mathcal{P}'$ be a set of representative patterns with non-empty cover sets found using the approximation strategy. Then we have (1) $\mathcal{P}' \subseteq \mathcal{P}$ and $\forall P \in \mathcal{P}'$, $|set(P)| \geq |set'(P)|$, where $set'(P)$ represents the cover set generated by the approximation strategy. (2) let $\mathcal{R}$ and $\mathcal{R}'$ be the minimum sets of RCPs generated from $\mathcal{P}$ and $\mathcal{P}'$, respectively, i.e., $\mathcal{R} \subseteq \mathcal{P}$ and $\mathcal{R}' \subseteq \mathcal{P}'$, we have $|\mathcal{R}| \leq |\mathcal{R}'|$.*

Proof. The first conclusion can be proved easily since the approximation strategy removes prevalent patterns from the cover set of a representative pattern. Therefore, for the same representative pattern P, $|set(P)| \geq |set'(P)|$. Consequently, if a representative pattern has non-empty approximate cover set (e.g., $|set'(P)| > 0$), its real cover set must be non-empty (e.g., $|set(P)| > 0$). That is, $\forall P \in \mathcal{P}'$, $P \in P$. However, the other way around may not be true. Thus, we have $\mathcal{P}' \subseteq \mathcal{P}$.

To prove the second conclusion, let's assume first $|\mathcal{R}'| < |\mathcal{R}|$. Since $\mathcal{R}' \subseteq \mathcal{P}'$ and $\mathcal{P}' \subseteq \mathcal{P}$, we must be able to find the same result set $\mathcal{R}'$ from

Algorithm 3.3 gen_cover_set(F_r, F, dis)

Input: F_r: the current representative pattern; F: a sub-pattern; dis: accumulated distance
Output: $\mathcal{S}$: all prevalent patterns ϵ-covered by F_r

```
1: for all P ⊂ F s.t. |F| − |P| = 1 & PI(P) ≥ minpi do
2:    dis = dis + TableLookup(P, F)
3:    if dis ≤ ε then
4:       Insert P to S
5:       gen_cover_set(F_r, P, dis)
6:    else
7:       dis = D(F_r, P)
8:       if dis ≤ ε then
9:          Insert P to S
10:         gen_cover_set(F_r, P, dis)
11: return S
```

$\mathcal{P}$. That is $\mathcal{R}' \subset \mathcal{P}$. In this case $\mathcal{R} = \mathcal{R}'$, which contradicts with $|\mathcal{R}'| < |\mathcal{R}|$. Hence the assumption is wrong. □

We will investigate the efficiency improvement gained by this approximate strategy and the incurred loss of compression rate in Section 3.6.

3.5.3 The *gen_cover_set()* Function

Integrating the optimization strategy and the approximation strategy discussed above, we present the function gen_cover_set() in Algorithm 3.3.

Given the input representative pattern F_r, Algorithm 3.3 visits its sub-patterns using a depth-first search. Line 1 finds all child prevalent co-location patterns of the current pattern F. Lines 2-4 implement the optimization strategy, when the co-location distance between F_r and a sub-pattern can be inferred to be smaller than ϵ. Otherwise, we have to compute the co-location distance (line 7). If the co-location distance is smaller than ϵ, we check further descendent sub-patterns (lines 9 -10). If not, the depth-first search can be stopped according to the approximation strategy.

3.6 Experimental Study

We have conducted comprehensive experiments to evaluate the proposed algorithms from multiple perspectives on both synthetic and real data sets. All algorithms are implemented in Python 2.7. All experiments are run on a PC with Intel Core Xeon 2.9 GHz CPU and 8 GB memory.

Parameters	Description	SynData_1	SynData_2	SynData_3
N_{seed}	The number of seed co-locations	5	5	50
λ	The parameter of Poisson distribution to define the number of instances of each seed co-location	1000	10000	1000
N_{aux}	The number of features added to construct auxiliary co-locations	3	5	5
α	The ratio that determines the number of instances of additional features when constructing auxiliary co-locations	0.7	0.9	0.9
r_{noise}	The ratio of the number of noise features over the number of features involved in seed and auxiliary co-locations	0.5	0.5	0.5
n_{noise}	The number of noise instances per noise feature	1000	1000	1000
$D_1 \times D_2$	The size of global spatial map	$10^5 \times 10^5$		
τ	The spatial distance threshold	10		
ϵ	The default value of co-location distance threshold	0.2		
$minpi$	The default value of prevalence threshold	0.4		

Table 3.2: Parameters used in synthetic data generation.

3.6.1 Experiments on Synthetic Data

In this section, we first introduce the synthetic data generator used in our experiments. Then we present the evaluation results on three different synthetic data sets.

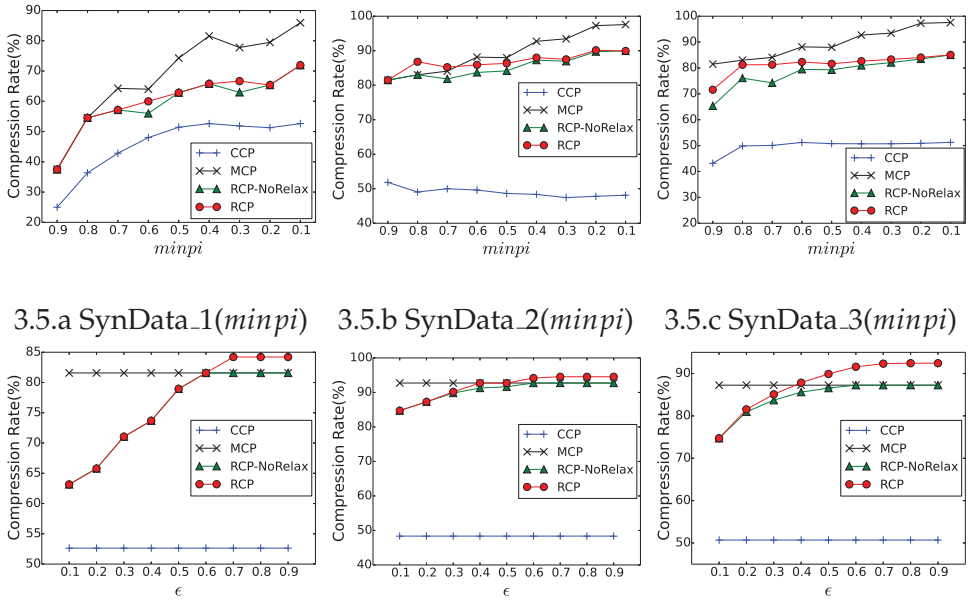
3.6.1.1 Synthetic Data Generator

Our synthetic data generation methodology is similar to the one used in [22] for co-location mining. Table 3.2 summarizes the parameters used in the generator. The data generation process begins with the generation of a set of N_{seed} seed co-locations. For each seed co-location, we randomly pick 2 features without replacement from a feature set. Next, we generate the instances of seed co-locations. The number of instances of a seed co-location is decided by a Poisson distribution with mean λ. To decide the positions of a seed co-location instances, we randomly pick a location (x_c, y_c) from the whole map $D_1 \times D_2$ as the center and set the radius as $r = \text{uniform}(0, \frac{\tau}{2})$, where $\text{uniform}(0, \frac{\tau}{2})$ selects a value from $(0, \frac{\tau}{2}]$ uniformly. Then we place instances within the circle. The coordinate of an instance is

$$(x_c + r \times \cos(\text{uniform}(0, 2\pi)), y_c + r \times \sin(\text{uniform}(0, 2\pi))).$$

After seed co-location instances are obtained, we then generate auxiliary co-locations by growing each seed co-location with N_{aux} additional features. Particularly, for each seed co-location with N_c instances and its corresponding circles, we randomly select $\alpha^i * N_c$ circles to insert an instance of the i^{th} additional feature, where $\alpha \in (0, 1]$ is the density ratio. A larger α leads to a denser data set. The final step is to generate noises, based on the two parameters of r_{noise}, the ratio of noise features, and n_{noise}, the number of noise instances per noise feature. Noise instances are placed randomly in the whole map.

Three synthetic data sets are generated with specific parameter values listed in Table 3.2. SynData_1 is a sparse data set while the other two are relatively dense. In particular, SynData_1 contains 37 features and 29,496 events; SynData_2 consists 52 features and 291,520 events, with more instances per co-location; and SynData_3 involves 525 features and 424,400 events.



3.5.a SynData_1($minpi$) 3.5.b SynData_2($minpi$) 3.5.c SynData_3($minpi$)

3.5.d SynData_1(w.r.t. ϵ) 3.5.e SynData_2(w.r.t. ϵ) 3.5.f SynData_3(w.r.t. ϵ)

Figure 3.5: Compression rate tests on synthetic data sets.

3.6.1.2 Compression Rate

We first evaluate the compression rate achieved by representative co-location pattern (RCP) mining, in comparison with closed co-location pattern (CCP) mining and maximal co-location pattern (MCP) mining. Specifically, we define *compression rate* as $(1 - \frac{N^*}{N_{PCP}}) \times 100\%$, where N^* equals to the number of compressed patterns and N_{PCP} refers to the number of prevalent co-location patterns (PCP).

Besides comparing with CCP and MCP, we also conduct experiments to investigate the compression rate of RCP without relaxation (RCP-NoRelax). As discussed in Section 3.3.3, we may either generate ϵ-clusters from the spatial data and return the prevalent centroid patterns as representatives, or relax the restriction to allow representative patterns to be prevalent w.r.t. $(1 - \epsilon) * minpi$ in order to achieve higher compression rate.

Figure 3.5 shows the compression rates of MCP, CCP, RCP, and RCP-NoRelax on the three synthetic data sets by varying the parameters $minpi$ and ϵ respectively. Overall, it can be observed that CCP has the lowest compression rate, while RCP achieves a higher compression rate than RCP-

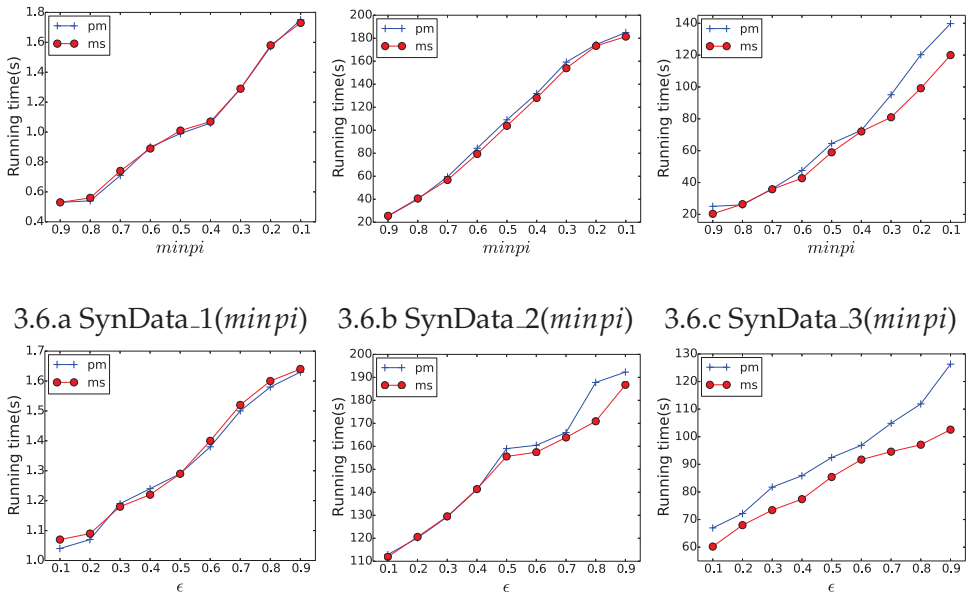
NoRelax. Regarding the comparison between RCP and MCP, we observe that MCP has a higher compression rate when ϵ is fixed at 0.2 (Figures 3.5.a, 3.5.b and 3.5.c). However, as ϵ is getting larger, RCP's compression rate prevails (Figures 3.5.d, 3.5.e and 3.5.f). That is, by relaxing the condition on the co-location distance threshold ϵ, RCP can achieve a compression rate which is even higher than that of MCP. This is due to the definition of RCP, while the best compression rate of RCP-NoRelax is bounded by that of MCP.

Moreover, it can be observed that RCP obtains a high compression rate on a dense data set. For example, when $\epsilon = 0.2$, the best compression rate of RCP on SynData_1 is 71.9% (Figure 3.5.a) while it is 89.9% and 85.0% on SynData_2 and SynData_3, respectively (Figure 3.5.b and Figure 3.5.c). This is because a representative pattern tends to cover more patterns on a dense data set. We also observe that when the prevalent threshold $minpi$ gets smaller, which means more co-location patterns are generated, the compression rate of RCP is higher. When the requirement on preserving the prevalence information is relaxed (i.e., when the co-location distance threshold ϵ is increased), the compression rate of RCP also improves, which is consistent with the definition of ϵ-cover relationship.

3.6.1.3 *RCPFast* vs. *RCPMS*

In this section we conduct experiments to compare the two proposed algorithms from different perspectives. Note that, all experiments are run 5 times and the average performance results are reported.

Framwork Comparison. Firstly, we compare the *post-mining* framework and the *mine-and-summarize* framework by implementing them without any optimization. That is, we implement *RCPFast* without the optimization based on *Theorem* 2 and *RCPMS* without the optimization strategy stated in Subsection 4.1 and the approximation strategy in Subsection 4.2. Figure 3.6 shows the running time with respect to the variation of $minpi$ and ϵ respectively. It can be seen that, on sparse data (SynData_1), the performance of the two frameworks is similar. This is because the *post-mining* framework performs quite fast on the sparse data set already. There is not much space for the *mine-and-summarize* framework to improve further. In contrast, when running on dense data sets (e.g., SynData_2 and SynData_3),



3.6.a SynData_1(*minpi*) 3.6.b SynData_2(*minpi*) 3.6.c SynData_3(*minpi*)

3.6.d SynData_1(w.r.t. ϵ) 3.6.e SynData_2(w.r.t. ϵ) 3.6.f SynData_3(w.r.t. ϵ)

Figure 3.6: Framework comparison on synthetic data sets.

the *post-mining* framework is relatively slower than the *mine-and-summarize* framework. The performance gap between the two frameworks enlarges on SynData_3 because SynData_3 is bigger than SynData_2, involving more features and events. Consequently, as we will show in Figure 3.8, there are more number of co-location distance computation required, which entangles the *post-mining* framework to spend more time on retrieving table instance information.

Computation Efficiency. We now compare the overall efficiency of the *RCPFast* algorithm and the *RCPMS* algorithm, both implemented with respective optimization strategies. In particular, we also implement a variation of *RCPMS*, called *RCPMS-NA*, which uses the optimization strategy in Subsection 4.1 only. Hence, by comparing *RCPMS* and *RCPMS-NA*, we can study the effectiveness of the approximation strategy in Subsection 4.2.

Figure 3.7 presents the running time on three synthetic data sets with respect to the variation of *minpi* and ϵ respectively. It can be observed that *RCPMS* outperforms *RCPFast* in all situations. Comparing the results on the three datasets, we note that the performance advantage of *RCPMS* is not as

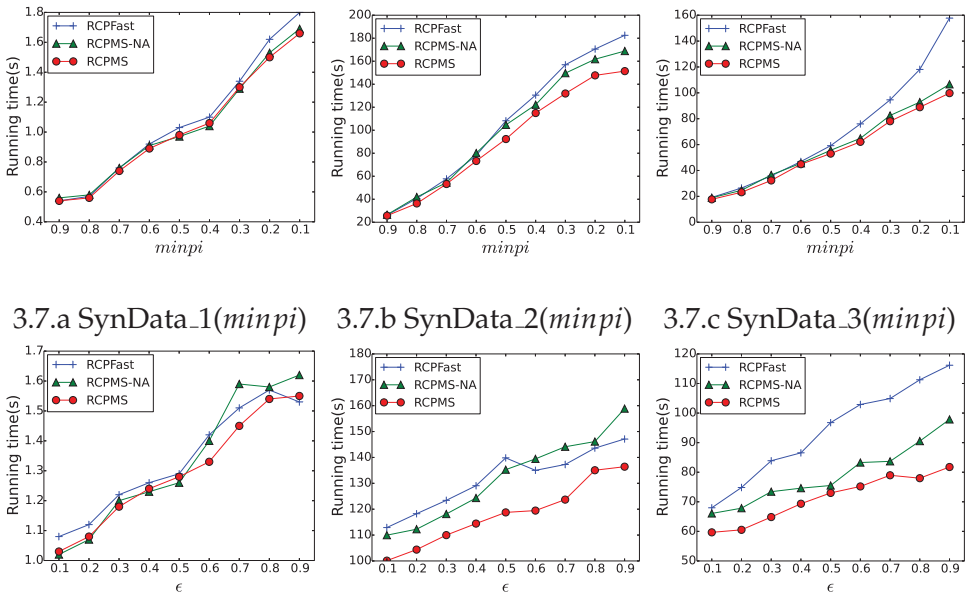
obvious on sparse data (SynData_1) as on dense data (SynData_2 and SynData_3). This is because when the data is sparse, the size of table instance of a co-location is small, resulting in a short time for computing co-location distance. Consequently, even if *RCPMS* reduces more number of co-location distance computation, the effect of computation saving of *RCPMS* is not obvious. The reasons for *RCPMS* being more efficient on the two dense data sets are different. For SynData_2, the data is dense in terms of the number of co-location instances, which leads to larger size of table instances and longer time to compute co-location distances. Specifically, the time of co-location distance computation is around 60ms on SynData_2 while it is 2.3ms and 3.5ms on SynData_1 and SynData_3, respectively [1]. Therefore, by reducing a few more number of co-location distance computation, *RCPMS* can show efficiency improvement clearly. SynData_3 is dense in terms of the number of co-locations. For this type of dense data, *RCPMS* demonstrates efficiency advantage by directly reducing the number of co-location distance computation.

By comparing *RCPMS* and *RCPMS-NA*, it is obvious, especially on SynData_2 and SynData_3, that the approximation strategy contributes significantly to the efficiency of the *RCPMS* algorithm.

Reduction of Co-location Distance Computation. The optimization strategies devised form both *RCPFast* and *RCPMS* aim to skip some co-location distance computation. To investigate the effectiveness of these strategies more thoroughly, we conduct experiments to record the number of co-location distance computations for *RCPFast*, *RCPMS* and *RCPMS-NA*, compared with the original number (baseline). Figure 3.8 presents the results by varying $minpi$ and ϵ respectively. It can be observed that all algorithms involves fewer co-location distance computations than the baseline does and the number of computations in *RCPMS* is the smallest.

In addition, by comparing *RCPFast* against the baseline, we notice that *Theorem 2* does reduce the number of co-location distance computations. However, when the reduced number is not great enough (e.g., Figures 3.8.a, 3.8.b, and 3.8.c), *Theorem 2* cannot contribute much to the performance im-

[1]The results are obtained by a profile tool for Python, available at https://github.com/rkern/line_profiler



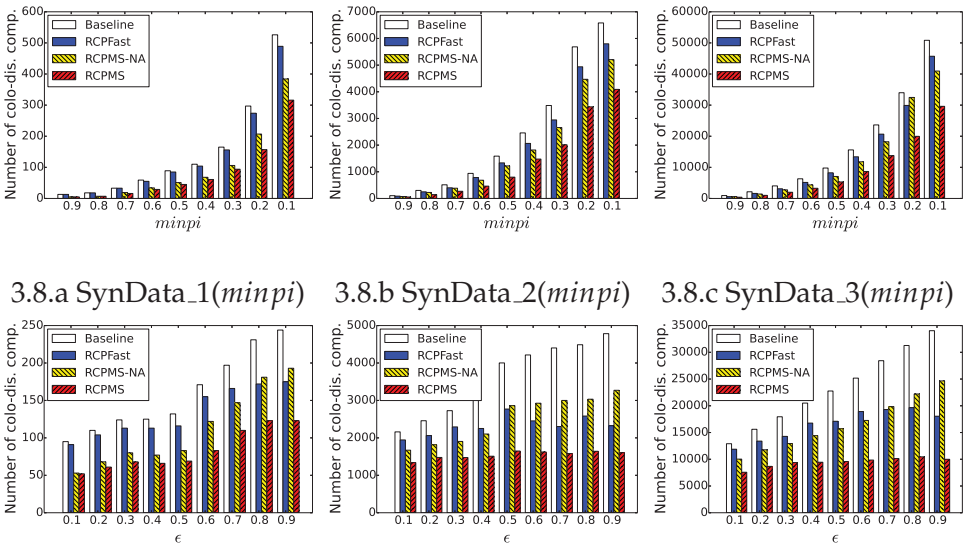
3.7.a SynData_1($minpi$) 3.7.b SynData_2($minpi$) 3.7.c SynData_3($minpi$)

3.7.d SynData_1(w.r.t. ϵ) 3.7.e SynData_2(w.r.t. ϵ) 3.7.f SynData_3(w.r.t. ϵ)

Figure 3.7: Performance tests with $minpi$ and ϵ on synthetic data sets.

provement. For example, the running times of *RCPFast* in Figures 3.7.a, 3.7.b, and 3.7.c is similar to those of the *post-mining* framework in Figures 3.6.a, 3.6.b, and 3.6.c. This is because it costs extra time for *RCPFast* to find all patterns that can be skipped according to *Theorem 2*.

Compression Rate. Although both Figures 3.7 and 3.8 show that the approximation strategy significantly improves the efficiency of *RCPMS*, as indicated by Lemma 3.2, more RCPs will be discovered by *RCPMS* than *RCPFast*. Hence, we further carry out experiments to evaluate how many more RCPs will be produced by *RCPMS*. We present the results using *compression rate difference*, which is calculated as $\frac{N_M - N_F}{N_{PCP}} \times 100\%$, where N_M and N_F refer to the numbers of patterns output by *RCPMS* and *RCPFast*, respectively. The results in Figure 3.9 show that the compression rate difference is less than 5% on all three datasets, regardless of the variation of parameters. Hence, *RCPMS* effectively improves the computation efficiency by sacrificing the compression rate very slightly.



3.8.a SynData_1($minpi$) 3.8.b SynData_2($minpi$) 3.8.c SynData_3($minpi$)

3.8.d SynData_1(w.r.t. ϵ) 3.8.e SynData_2(w.r.t. ϵ) 3.8.f SynData_3(w.r.t. ϵ)

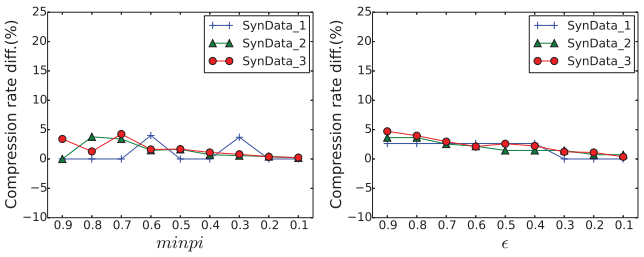
Figure 3.8: Co-location distance computation analysis on synthetic data sets.

3.6.2 Experiments on Real Data

Two real-world data sets are used in our experiments. The first one is from the EPA databases, which contain environmental activities that affect air, land and water in United States[2]. Different environmental interest types are used as spatial features and each facility represents a spatial event. In our experiment, we use the EPA data of Allen Counties in Indiana State, which consists of 23 features and 647 events in total. The second data set is the points of interest (POI) in California[3], which was used in [59]. There are 63 category types (e.g., dam, school, and bridge) and 104,770 data points. All the geographic coordinates are transformed to 2-dimensional Cartesian coordinates by Universal Transverse Mercator projection. The spatial distance threshold is 2000 by default (meaning 2km in real world).

We first investigate the compression rate of RCP mining. Figure 3.10 illustrates the compression rate of *RCPFast* and *RCPMS* on the two real data sets by varying $minpi$ and ϵ respectively. We set the default values as

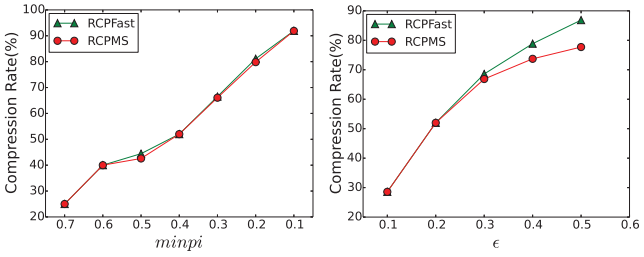
[2]http://www.epa.gov/
[3]http://www.usgs.gov/



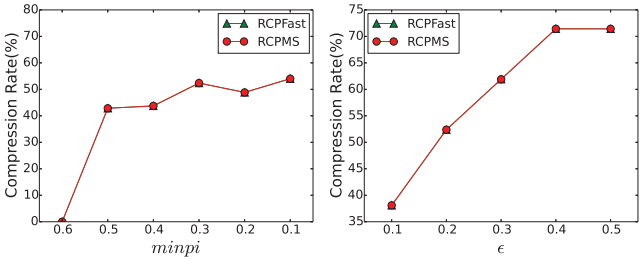
3.9.a w.r.t. $minpi$ 3.9.b w.r.t. ϵ

Figure 3.9: Compression rate differences between *RCPMS* and *RCPFast* on synthetic data sets.

$minpi = 0.4$ and $\epsilon = 0.2$. Generally, the compression rates of the two algorithms are close to each other, except on the EPA dataset when ϵ is large (e.g., Figure 3.10.b where $\epsilon = 0.5$). However, in that situation, we note *RCPMS* still can reach a compression rate as high as 75%, which is acceptable. Also it can be observed that the compression rates increase when $minpi$ is decreased or ϵ is increased, which is consistent with the results obtained from the synthetic data sets.



3.10.a EPA (w.r.t. $minpi$) 3.10.b EPA (w.r.t. ϵ)

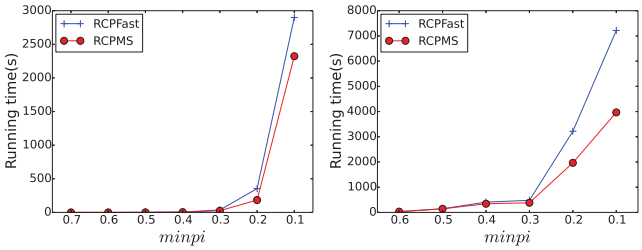


3.10.c POI (w.r.t. $minpi$) 3.10.d POI (w.r.t. ϵ)

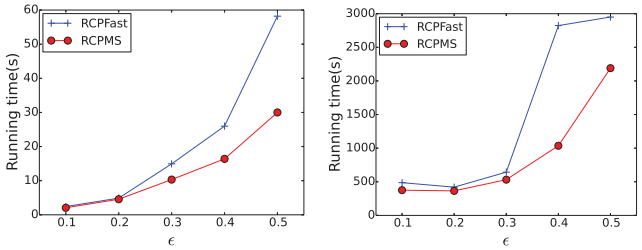
Figure 3.10: Compression rate tests on EPA and POI data sets.

Next, we study the efficiency of the proposed algorithms on real data

sets. Figure 3.11 illustrates the running time of the two algorithms with respect to the variation of $minpi$ and ϵ respectively. It shows that *RCPMS* outperforms *RCPFast* on the two real datasets, especially when the data is getting dense (e.g., when $minpi$ is decreased) or the requirement of preserving prevalence information is relaxed (e.g., when ϵ is increased).



3.11.a EPA (w.r.t. $minpi$) 3.11.b POI (w.r.t. $minpi$)



3.11.c EPA (w.r.t. ϵ) 3.11.d POI (w.r.t. ϵ)

Figure 3.11: Performance on EPA and POI data sets.

To examine whether the summarized representative co-location patterns are meaningful, we also inspect the discovered patterns. We show the experimental results on the POI data set when $minpi = 0.6$ and $\epsilon = 0.2$ as an example. In this case, seven prevalent co-location patterns are discovered: BUILDING-PARK, CHURCH-PARK, PARK-SCHOOL, BUILDING-PO(*abbreviation of post office*), BUILDING-SCHOOL, BUILDING-CHURCH, and BUILDING-PARK-SCHOOL. Each pattern refers to a set of points of interests that are frequently located in proximity. By running the *RCPMS* algorithm on the data, we discover four representative co-location patterns: BUILDING-PARK-SCHOOL, BUILDING-PO, PARK-SCHOOL and BUILDING-CHURCH-PARK. It can be seen that it makes sense to use the compressed patterns to represent the original patterns. Moreover, among the four RCPs, only the pattern BUILDING-CHURCH-PARK is not a prevalent co-location. However, the PI value of

BUILDING-CHURCH-PARK is 0.51, which is greater than $(1 - 0.2) \times 0.6 = 0.48$.

3.7 Conclusions

In this chapter, we study the problem of summarizing spatial co-locations using representative patterns. Addressing the missing of transactions in spatial data, a new measure is defined to appropriately quantify the prevalence distance between two co-location patterns, based on which the problem of representative co-location pattern mining is formulated. We propose two efficient algorithms for RCP mining: *RCPFast* and *RCPMS*. *RCPFast* follows existing approaches to adopt a *post-mining* framework that finds representative patterns from the set of discovered prevalent co-location patterns. *RCPMS* employs a novel *mine-and-summarize* paradigm to discover representative patterns directly from the spatial data set, thereby pushing pattern summarization into the co-location mining process. Experimental results show that RCP mining effectively summarizes prevalent co-location patterns, and *RCPMS* significantly improves over *RCPFast* on dense data sets by slightly sacrificing compression rate. Moreover, the proposed summarizing method is not only helpful in static co-location pattern mining, but also useful in the dynamic scenario, *e.g.*, time-evolving co-location pattern mining. For the dynamic case, we can aggregate the representative patterns within different time window to extract the interesting patterns.

Chapter 4

Protecting Location Privacy in Spatial Crowdsourcing

Spatial crowdsourcing (SC) has emerged as a transformative platform providing services to outsource spatio-temporal tasks to a set of registered individual workers, who may be willing to physically travel to locations of interest and perform tasks. In order to assign tasks to workers in proximity of the task locations, existing SC solutions usually require workers to disclose their locations, which inevitably raises privacy concerns about the workers' locations. Although a handful of methods have been proposed to protect location privacy using cloaking techniques or random perturbation, we argue that it is more secure for workers to encrypt their location information when they register with the SC platforms. In this chapter, we propose a secure SC framework based on encryption, that ensures that workers' location information is never released to any party, yet the system can still assign tasks to workers situated in proximity of each task's location. We solve the challenge of assigning tasks based on encrypted data using *homomorphic encryption*. Moreover, to overcome the efficiency issue, we propose a novel secure indexing technique with a newly devised *SKD-tree* to index encrypted worker locations. Experiments on real-world data evaluate various aspects of the performance of the proposed SC platform.

4.1 Introduction

Crowdsourcing, using 'wisdom of the crowd' or the knowledge and opinion of a group of individuals, has been widely adopted to outsource manual tasks, such as image labeling or natural language understanding, to the public for quick services. With the pervasiveness of mobile devices, the ubiquity of wireless network and the improvement of sensing technology, a new mode of crowdsourcing, namely *Spatial Crowdsourcing (SC)*, has emerged [75, 76]. In SC, *task requesters* register through a centralized *spatial crowdsourcing server* (*SC-server*) and request resources related to tasks situated in specific locations. The SC server assigns tasks to registered *workers* according to performance criteria. If a worker accepts the assigned task, he/she physically travels to the location to perform the required task. For example, in traffic management, SC allows engaged individuals with mobile phones to report nearby traffic conditions so that effective traffic control can be used to alleviate traffic jam situations. Many other SC applications can be found in a variety of domains, such as weather monitoring, environmental sensing, crisis response, and urban planning.

In order to avoid long distance travel for workers (*i.e.*, to minimize the costs), existing SC systems usually require workers to disclose their location in the form of either spatial points or approximate regions [77]. In reality, SC servers may not be fully trustworthy, so disclosing individual locations may have serious privacy implications. For example, with the leakage of workers' location information, an adversary may invoke a broad spectrum of attacks such as physical stalking, identity theft, and breach of sensitive information including health status, political or religious views [78]. Location privacy is therefore a critical privacy issue and it is important to develop secure SC frameworks to ensure maximum security.

Several approaches have been proposed [79] to protect workers' locations using a trusted third party (TTP). In their framework, each worker subscribes to a *cellular service provider* (*CSP*), which serves as the TTP and has access to all worker locations. When assigning tasks, the CSP releases location information to the SC server in a noisy form, and the SC server then queries sanitized data and disseminates tasks to workers. One major

drawback of this approach is that there is no privacy protection with respect to the CSP. Once CSP is compromised by adversaries, location privacy is infringed. Not to mention that this framework does not protect task locations.

Alternatively, a TTP-free privacy-preserving framework [78] can be implemented by obfuscating with each worker's location as a probabilistic distribution, as opposed to a deterministic value. The goal of the SC-server is to assign as many tasks as possible based on approximate information. Unfortunately, by simply observing location distributions, the SC server is able to approximately guess a worker's location. In addition, the server knows the final task assignment results, from which it can infer worker locations with reasonable confidence. For example, if the SC server knows that a number of tasks have been assigned to a particular worker, it can draw circles using task locations as centers and the maximum travel distance as the radius, which will easily conclude, with high probability, that the worker is located at the intersection of those circles.

The above observations motivate our work, which aims to deliver a general trustworthy SC framework with improved privacy by requiring workers and requesters to encrypt their location data when registering with and exchanging information through the SC server. Using the correct settings and protocols, real location information is hidden in the ciphertexts and is never disclosed to any party. Accordingly, we can deliver a secure SC framework to sufficiently preserve the location privacy of both workers and requesters based on encryption.

Although encryption provides maximum security protection, the challenge is that the SC server has to assign tasks (*e.g.*, compute the distance between tasks and workers) based on encrypted data. We solve the problem of computation on ciphertexts with a *homomorphic encryption* scheme and introduce a dual SC server design. To address the inefficiency of encryption operations, we propose a secure indexing technique with a newly devised *SKD-tree* to index encrypted worker locations for fast searching and pruning.

We have named our SC framework HESI, as it combines a homomorphic encryption (HE) scheme and a secure indexing (SI) technique. The proposed framework has the following properties:

1. *Location privacy*: The HE scheme ensures zero worker location privacy leakage. A worker's location is never disclosed to any parties except themselves. Task locations are also preserved and are only exposed to the assigned workers.

2. *Data privacy*: Task information is only disclosed to a small group of workers who will potentially perform the tasks.

3. *Computation efficiency*: An SI technique improves the efficiency and scalability of the task assignment process. The proposed SKD-tree also allows efficient operations such as insertion and search.

4. *Lightweight computation on the client side*: The workers and requesters are only required to perform a small number of encryptions/decryptions.

The remainder of the chapter is organized as follows. Related works are reviewed in Section 4.2. Section 4.3 discusses relevant research prelimiaries. Section 4.4 presents the architecture of the HESI framework and describes the overall workflow. Secure distance computation, secure indexing and task assignment are discussed in Section 4.5, Section 4.6, and Section 4.7 respectively. Section 4.8 analyzes the security and complexity. Experiments are reported in Section 4.9, and we conclude this chapter in Section 4.10.

4.2 Related Works

In this section, we briefly provide an overview of existing works on location privacy, task assignment in spatial crowdsourcing and secure index.

4.2.1 Location Privacy

Location privacy is a key factor in many real-world applications, such as sensor network [80, 81], social network [82],cloud computing [83], mobile computing [84, 85], and Internet of Things (IoT) [86]. Existing methods for location privacy protection roughly fall into two categories. Methods in the first category rely on a trusted third party (TTP) to control privacy leakage while preserving utility. In this context, a technique of k-anonymity

is commonly adopted, where the real location of a user is replaced by a cloaking area which guarantees that at least k users are located [87]. Many extended works have been studied based on k-anonymity, such as [88, 89], in which the personal privacy requirement can be defined by individual users, and [90], in which the cloaking techniques are based on grids. Location perturbation techniques are also proposed to protect object locations [91] [92] [93]. However, there is a potential privacy hazard in relying on a TTP. The second category of location privacy preserving methods run without a TTP. Three representative TTP-free methods are assessed in [94]. The first is *collaboration-based*, in which a user constructs a centroid location area with other users [95] [96]. The second is *obfuscation-based*, where the real location of a user is usually replaced by a circular area of variable center and radius [97] [26]. The last method is *cryptography-based*. One common technique is the Private Information Retrieval (PIR) protocol that allows a user to retrieve information from a server without revealing what information has been retrieved. It has been applied to solving secure kNN problems [98] [99].

Many works have been proposed to protect location privacy in spatial crowdsourcing [100, 89]. [79] proposes a framework where a *cellular service provider* (CSP) acts as a TTP holding all worker locations. It publishes their locations to the SC server using the *Private Spatial Decomposition* approach [101]. This framework suffers from the same problems as TTP methods. Moreover, the requester/task locations are not protected. A TTP-free privacy-preserving framework that protects worker location privacy was studied in [78]. It is based on the obfuscation method where each worker location is considered as a probabilistic distribution. However, the SC server has the knowledge of location distribution and assignment results, which leads to a security threat whereby the server is able to deduce a worker location with high probability.

4.2.2 Secure Index

Several secure index techniques exist to facilitate an efficient query process [102, 103, 104]. In [105], Hore et al. built an index by partitioning the data

into a set of buckets, enabling the data owner to index for buckets, outsourcing all data to the cloud, but retained the indexes at his/her own site. In [106], an encrypted traversal framework based on privacy homomorphism was proposed to process private queries. A multi-level indexing data structure based on R-tree is provided to securely assess the distance information between two data points. Wang et al. proposed a hierarchical encrypted index model to execute range queries on outsourced databases [107]. They designed a secure $\hat{R}$-tree based on Asymmetric Scalar-product Preserving Encryption (ASPE) [108], where encrypted data points carry information relating to their distance from the origin. As an extended work, Cheng et al. proposed a new encryption method, ASPE with Laplace noise, and used an SR-tree (secure R-tree) to enable secure kNN query processing over encrypted spatial data [109]. All these previous works require the client (or data owner) to take part in the computation process. This means that the client side must have sufficient computation resources; however, this is not always the case, especially in the mobile sensing scenario. Our secure index technique is different from previous work because in our setting, the index processing is done on the server side, alleviating the computation burden on the data owner.

4.2.3 Task Assignment in SC

Many studies on task assignments in spatial crowdsourcing have been proposed [110] [111] [112, 113, 114, 115]. In [75], it is assumed that if a worker receives the task, he will certainly complete it. Hence, only one worker is needed for each task. Based on this work, a new constraint whereby the workers are not trustworthy is considered in [116]. Each worker is associated with the probability that the worker will perform a task; consequently, the success rate of task assignment becomes probabilistic. In [110], it is assumed that each task consists of several sub-tasks. A task is completed only if all sub-tasks are completed. It is proved that this problem is reducible to the maximum flow problem. In [117], it is assumed that a task requires the expertise type (e.g., professional photographer) and degree of expertise. Our framework produces a distance matrix between tasks and workers and

can directly (or easily be extended to) support most task assignment models.

4.3 Preliminariy

In this section, we introduce necessary background on spatial crowdsourcing, privacy concerns, and the Paillier cryptosystem.

4.3.1 Spatial Crowdsourcing Model

One of the important process in spatial crowdsourcing is to assign spatial tasks to workers, who are willing to perform tasks at specific locations. According to [116], spatial task publishing modes, *i.e.*, the ways in which workers are matched to tasks, can be categorized into two types: Worker Selected Tasks (WST) and Server Assigned Tasks (SAT).

In the WST mode, an SC server publishes the spatial tasks online and workers can autonomously choose any task in their proximity without the need to coordinate with the SC server. A clear advantage of this mode is that workers do not need to reveal their location to the SC server, and thus preserve their location privacy. However, the ultimate assignment result is usually sub-optimal since workers do not have a global view. It may cause a situation in which multiple workers choose the same task while other tasks remain unassigned. In SAT mode, workers share their locations with the SC server which is in charge of assigning tasks to workers according to certain goals. Because the SC server knows all the spatial information, SAT can achieve global optimization by running delicate matching algorithms, but it requires workers to disclose their location to the server, which raises a privacy issue.

In our work, we focus on the SAT mode. To remedy the privacy issue, we provide location privacy protection instead of directly revealing workers' locations, and allow workers to encrypt their location before sending it to the server.

4.3.2 Threat Model

Similar to existing work on secure outsourcing [99][118], we adopt settings and definitions commonly used in secure multi-party computation (SMC) [119]. In this setting, there are two main types of models: *semi-honest* and *malicious* [120]. In the semi-honest model all parties (including the corrupted ones) follow the protocol rules, but are later free to use what they see during the execution of the protocol to compromise security. The semi-honest parties are also known as "honest-but-curious" and "passive". In comparison, malicious parties, called "active", can arbitrarily manipulate the protocol specification according to an adversary's instruction. In practice, the malicious model does exist but is too inefficient to be used or implemented. However, the semi-honest model is not only useful but also the foundation of designing secure protocols in the malicious model. Therefore, following existing work in [99], [121] and [122], we have adopted the semi-honest model for our framework.

We assume that all encryptions are secure and no adversary is able to derive plaintext from ciphertext without the correct key. Considering the power of the adversary, we allow the adversary to have knowledge of some plaintext and the corresponding ciphertext in advance (also known as a known-ciphertext attack [123]).

The security of a protocol in the semi-honest model is analyzed using a simulation-based proof technique [119]. The key idea of this technique is to construct a simulator to show that the real protocol behaves like some idealized ones. Since essentially no attacks can be carried out in the ideal world, security is implied. The details of the security analysis are discussed in Section 4.8.

4.3.3 Paillier Cryptosystem

The Paillier cryptosystem is an additive homomorphic and multiplicative homomorphic encryption scheme [124]. It is commonly used in SMC applications such as electronic voting and electronic cash. Let E_{pk_C} be the encryption function with public key pk given by (N, g) where N is a product of two large primes and g is in $\mathbb{Z}^*_{N^2}$. Given $a, b \in \mathbb{Z}_{\mathbb{N}}$, the Paillier encryption

scheme has the following properties:

1. **Homomorphic Addition**

$$E_{pk}(a+b) = E_{pk}(a) * E_{pk}(b) \bmod N^2 \tag{4.1}$$

2. **Homomorphic Multiplication**

$$E_{pk}(a*b) = E_{pk}(a)^b \bmod N^2 \tag{4.2}$$

That is, we can perform addition or multiplication of plaintexts by manipulating ciphertexts with multiplication or exponentiation, respectively. In our framework, all encryptions are based on the Paillier cryptosystem.

4.4 The HESI Framework

We present an overview of the proposed framework HESI in this section, including the system architecture and the overall workflow.

4.4.1 The Dual-Server Architecture

Our secure SC problem can be considered as a secure outsourcing multi-party computation [125]. Our aim is to enable the SC server to carry out all the computations while users (workers/requesters) do nothing but perform a small number of encryptions and decryptions.

In a single server setting, each user outsources their encrypted data with their private key to the server and the server is able to perform computations on encrypted data using homomorphic encryption (HE). Multi-key homomorphic encryption (MHE) seems to be an optimal solution to achieve feasibility [126]. However, a user will need to recover the results interactively by participating in another SMC protocol, which is very inefficient in terms of communication and computation. Therefore, the goal of lightweight computation, on the user side, is difficult to achieve in a *multi-user*, *one-server* setting.

To solve this challenge, we have adopted a *dual-server* design and propose an SC framework that consists of two non-colluding semi-honest servers. The assumption of non-collusion between two service providers, such as Google and Amazon, is reasonable in practice [121], because the collusion of two well-established companies may damage their reputation and consequently reduce their revenues.

According to the semi-honest model, these two servers are curious but will follow the protocols. A dual-assisted server setting can liberate users from heavy computation and communication by allowing the server to complete computation tasks. The security intuition behind dual-server settings has been addressed in related domains such as secure multi-party computation [127], secure kNN search [118] and secure trajectory computation [122], and we refer interested readers to these texts for further rationale.

Figure 4.1 illustrates the HESI framework. It consists of workers, requesters, and two servers: the *logistics server* (S_L) and the encrypted data *computing server* (S_C). In general, the requesters submit their tasks to the SC platform and the tasks are dispatched to appropriate workers. The S_L handles all logistics issues, including new user/task registration, data indexing maintenance (*i.e.*, SKD tree), and task assignment, whereas the computing server is an auxiliary server that handles the computation of encrypted data. We assume that each server owns a pair of encryption keys, *i.e.*, (pk_L, sk_L) and (pk_C, sk_C), where pk is the public key and the private key sk is known only to owner.

4.4.2 The System Workflow

Let $\mathcal{W}$ and $\mathcal{T}$ denote the set of workers and tasks, respectively. Each worker only accepts a limited number of tasks at the same time and accepts only those within a certain distance. In the following, we assume that all locations are 2-dimensional, yet our framework can be easily extended to multiple dimensions. The whole process is presented as follows:

(1) Worker Registration (WR). Workers register and sign up with the S_L.

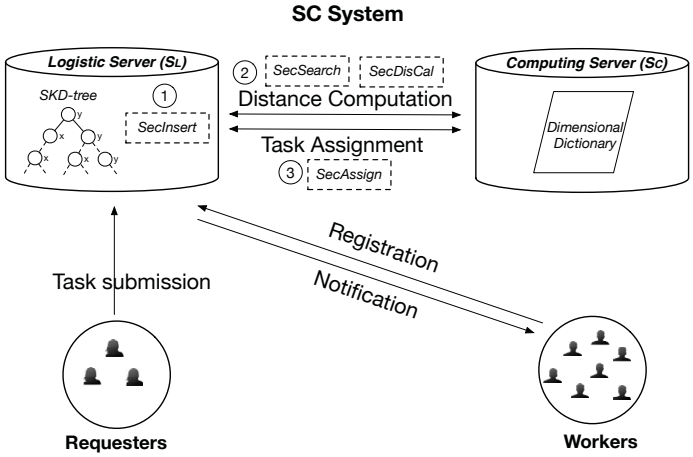


Figure 4.1: The HESI framework.

Each worker's registration information is a tuple:

$$< w_i, E_{pk_C}(l_{w_i}), T_i, D_i >,$$

where w_i is the worker identity, l_{w_i} is the worker's current location, E_{pk_C} represents the Paillier encryption, T_i refers to the maximum number of tasks that a worker is willing to accept at one time, and D_i refers to the acceptable maximum travel distance. The S_L will index all workers' encrypted locations using an interactive secure indexing (SI) technique and store them in a data structure called an SKD-tree, as shown in Figure 4.1. The construction of the SKD-tree and the usage of the tree for fast range search is discussed in Section 4.6. The D_i information, necessary at the task assignment stage, is also transferred to the S_C.

(2) Task Submission (TS). Apart from worker locations, our framework also protects the privacy of requester/task locations. The requesters submit their tasks to the S_L in the following form:

$$< t_i, E_{pk_C}(l_{t_i}), reward, content >,$$

where t_i is the task identity, l_{t_i} represents the task location, *reward* refers to the payment for the workers and *content* specifies the task mission, *e.g.*, taking photos or reporting traffic conditions at the location l_{t_i}.

(3) Distance Computation (DC). In order to assign workers to tasks in

close proximity, the SC platform needs to know the distances between tasks and workers. The S_L and S_C together perform an interactive protocol to compute the distance based on the encrypted data using homomorphic encryption. The distance between tasks and workers can be used for evaluation during task assignment, while the real locations of tasks and workers are never revealed to either the S_L or S_C. In addition, neither the S_L and S_C are able to learn any sensitive information from the intermediate result, unless they conspire which is not allowed according to the protocol. The details of the interactive protocol for distance computation are discussed in Section 4.5.

(4) Task Assignment (TA). Based on a distance matrix $\mathcal{M}$, where element m_{ij} represents the distance between t_i and w_j, and the task acceptance conditions of workers (*e.g.*, T_i and D_i), the SC-system assigns the tasks to workers with the goal of maximizing the number of assigned tasks while minimizing the workers' travel costs. The task assignment is carried out by an interactive protocol between the S_L and the S_C, under conditions that both servers cannot learn any sensitive information from the intermediate results. The details of task assignment strategy are discussed in Section 4.7. The task assignment results are preserved in the form $< t_i : w_1, \ldots, w_k >$, indicating that task t_i is assigned to $w_1, \ldots, w_k$.

(5) Task Notification (TN). The final stage is for the S_L to notify assignment results to corresponding workers. Because the S_L does not know the exact locations of the tasks, it needs to communicate with requesters as follows. The S_L first sends the workers' public keys to the requester according to the assignment results. For example, if a result record is $< t_i : w_1, \ldots, w_k >$, t_i will receive the corresponding workers' public keys. Next, the requester encrypts the task's location with the received public keys and sends them back to the S_L, in the form $< (w_1, E_{pk_{w_1}}(l_{t_i})), \ldots, (w_k, E_{pk_{w_k}}(l_{t_i})) >$. The S_L then notifies the worker w_j with a message $< (E_{pk_{w_j}}(l_{t_1}), content_1), \ldots, (E_{pk_{w_j}}(l_{t_q}), content_q) >$. When w_j receives the message, she decrypts the ciphertext using her own private key sk_{w_j} to obtain the task's location. The worker is then able to travel to the specified location and perform the task according to the mission described in its *content*.

Implementation details are discussed in the following sections, includ-

Table 4.1: The outline of the secure protocols.

Scope	Protocols	Description
Index	*SecInsert*	Insert a node (worker's encrypted location) into an SKD-tree securely.
	SecVerifySide	Verify the insertion side with respect to a given node. An auxiliary protocol of *SecInsert*.
	SecSearch	Given a spatial range, output a set of workers within this range.
	SecComp	Compare two integers securely. An auxiliary protocol of *SecSearch*.
Computation	*SecDisCal*	Calculate the distance of two locations securely.
	SecMul	Compute the multiplication of two integers securely. An auxiliary protocol of *SecDisCal*.
Assignment	*SecAssign*	Perform secure task assignment according to some strategies.

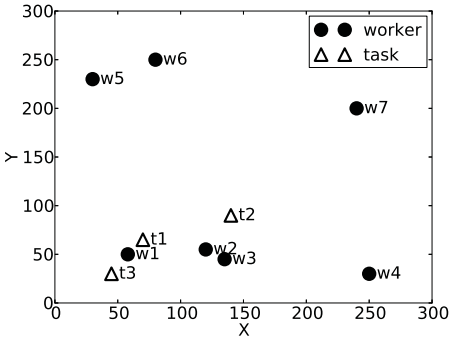
ing the secure distance computation (Section 4.5), secure indexing (Section 4.6), and the task assignment solution (Section 4.7).

For clarity, we briefly outline our proposed secure protocols in Table 4.1. These secure protocols are categorized into distance computation, secure index and assignment. *SecDisCal*, *SecInsert*, *SecSearch SecAssign* are the main protocols, while the others are auxiliary protocols.

We have used the small dataset shown in Figure 4.2 as a 'toy' example for illustration. There are 3 tasks (denoted by triangles) and 7 workers (denoted by circles) in a 300×300 spatial region. All coordinate information is given in the table. The circle symbol denotes a task location and the triangle represents a worker location. The coordinates are given in the table.

4.5 Secure Distance Computation

Recall that the third stage of HESI is to compute the distances between tasks and workers. The challenge of this stage is to compute exact distances without knowing the real locations. The S_L possesses the encrypted locations (*e.g.*, $E_{pk_C}(l_{t_i})$ and $E_{pk_C}(l_{w_j})$), while the S_C owns the private key sk. The objective is to let the S_C derive the distance $d_{ij} = ||l_{t_i} - l_{w_j}||$ without knowing l_{t_i} and l_{w_j}. To solve the problem, we introduce a secure protocol *SecDisCal*



	X	Y
t_1	70	65
t_2	140	90
t_3	45	30
w_1	58	50
w_2	120	55
w_3	135	45
w_4	250	30
w_5	30	230
w_6	80	250
w_7	240	200

Figure 4.2: A small spatial dataset.

Algorithm 4.1 SecDisCal$(E_{pk_C}(l_1), E_{pk_C}(l_2))$

Input: S_L has $E_{pk_C}(l_1)$, $E_{pk_C}(l_2)$ and S_C has sk.
Output: S_C obtains $|l_1 - l_2|^2$.

1. S_L:
   (a) Compute $E_{pk_C}(l_{1_x} - l_{2_x}) = E_{pk_C}(l_{1_x}) * E_{pk_C}(l_{2_x})^{N-1}$
   (b) Compute $E_{pk_C}(l_{1_y} - l_{2_y}) = E_{pk_C}(l_{1_y}) * E_{pk_C}(l_{2_y})^{N-1}$
   (c) Send the identity of l_1 and l_2 to S_C.
2. S_L and S_C:
   Based on *SecMul* protocol, compute $d_x = E_{pk_C}((l_{1_x} - l_{2_x})^2)$ and $d_y = E_{pk_C}((l_{1_y} - l_{2_y})^2)$.
3. S_L:
   Compute $d = d_x * d_y = E_{pk_C}((l_{1_x} - l_{2_x})^2 + (l_{1_y} - l_{2_y})^2)$, which equals to $E_{pk_C}(|l_1 - l_2|^2)$.

as described in Algorithm 4.1, which interacts between the S_L and the S_C using homomorphic encryption (HE).

The inputs of *SecDisCal* include two encrypted locations, where each coordinate dimension is correspondingly encrypted using public key pk_C (*e.g.*, $E_{pk_C}(l_{1_x})$ and $E_{pk_C}(l_{1_y})$). In step 1, the S_L computes the subtraction of each dimension. Then the S_L and S_C compute the square of the subtraction using a protocol *SecMul*, which will be explained later. Lastly, S_L adds the squares using homomorphic encryption (*i.e.*, Eq.(4.1)) and obtains the encrypted square distance between l_1 and l_2. We note that while the locations in Algorithm 4.1 are two-dimensional, *SecDisCal* can be easily extended to multi-dimensional applications.

Algorithm 4.2 SecMul($E_{pk_C}(x_1), E_{pk_C}(x_2)$)

Input: S_L has $E_{pk_C}(x_1)$, $E_{pk_C}(x_2)$ and S_C has sk.
Output: $E_{pk_C}(x_1 * x_2)$.

1. S_L:
(a) Pick two random numbers $r_1, r_2 \in \mathbb{Z}_N$.
(b) Compute $x'_1 = E_{pk_C}(x_1) * E_{pk_C}(r_1)$, $x'_2 = E_{pk_C}(x_2) * E_{pk_C}(r_2)$ and send x'_1, x'_2 to S_C.
2. S_C:
(a) Decrypt x'_1, x'_2 and have $m_1 = D_{sk}(x'_1)$, $m_2 = D_{sk}(x'_2)$.
(b) Compute $m = m_1 * m_2 \mod N$.
(c) Compute $m' = E_{pk_C}(m)$ and send m' to S_L.
3. S_L:
(a) Compute $s'_1 = E_{pk_C}(x_1)^{N-r_2}$, $s'_2 = E_{pk_C}(x_2)^{N-r_1}$ and $s'_3 = E_{pk_C}(r_1 * r_2)^{N-1}$.
(b) Compute $((m' * s'_1) * s'_2) * s'_3$ step by step. The final result equals $E_{pk_C}(x_1 * x_2)$.

Figure 4.2 demonstrates the calculation of the distance between t_1 and w_1 using *SecDisCal*. Firstly, the S_L computes $E_{pk_C}(t_{1_x} - w_{1_x}) = E_{pk_C}(70 - 58) = E_{pk_C}(70) * E_{pk_C}(-58) = E_{pk_C}(12)$ and $E_{pk_C}(t_{1_y} - w_{1_y}) = E_{pk_C}(65 - 50) = E_{pk_C}(65) * E_{pk_C}(-50) = E_{pk_C}(15)$. Secondly, the square coordinate differences can be derived based on *SecMul*, *i.e.*, $d_x = E_{pk_C}(12^2)$ and $d_x = E_{pk_C}(15^2)$. Then, the S_L computes the square distance $d = E_{pk_C}(12^2 + 15^2) = E_{pk_C}(12^2) * E_{pk_C}(15^2) = E_{pk_C}(369)$.

Algorithm 4.2 illustrates the secure multiplication protocol *SecMul*. Given two encrypted integers $E_{pk_C}(x_1)$, $E_{pk_C}(x_2)$, the protocol *SecMul* outputs their encrypted multiplication $E_{pk_C}(x_1 * x_2)$. At step 1, the S_L perturbs the encrypted locations by adding noisy integers according to Eq. (4.1) (*i.e.*, $x'_1 = E_{pk_C}(x_1 + r_1)$ and $x'_2 = E_{pk_C}(x_2 + r_2)$). After step 2, the S_C has $m' = E_{pk_C}((x_1 + r_1) * (x_2 + r_2))$. By expanding the polynomial, we have $m' = E_{pk_C}(x_1 * x_2 + x_1 * r_2 + x_2 * r_1 + r_1 * r_2)$. Therefore at step 3, the S_L needs to eliminate the terms $x_1 * r_2$, $x_2 * r_1$ and $r_1 * r_2$ to obtain the desired output $E_{pk_C}(x_1 * x_2)$.

To give an example of computing the square coordinate differences, suppose the inputs of *SecMul* are $E_{pk_C}(x_1) = E_{pk_C}(x_2) = E_{pk_C}(12)$ and without lost of generality, let $r_1 = 1$, and $r_2 = 2$. Firstly, the S_L computes $x'_1 = E_{pk_C}(12) * E_{pk_C}(1) = E_{pk_C}(13)$, $x'_2 = E_{pk_C}(12) * E_{pk_C}(2) = E_{pk_C}(14)$

and sends them to the S_C. Next, the S_C decrypts the ciphertext and computes $m = m_1 * m_2 = 13 * 14 = 182$. Then it sends $m' = E_{pk_C}(182)$ to S_L. In the last step, the S_L computes $s'_1 = E_{pk_C}(12)^{N-2} = E_{pk_C}(-24)$, $s'_2 = E_{pk_C}(12)^{N-1} = E_{pk_C}(-12)$ and $s'_3 = E_{pk_C}(1 * 2)^{N-1} = E_{pk_C}(-2)$. The result is obtained by computing $E_{pk_C}(182) * E_{pk_C}(-24) * E_{pk_C}(-12) * E_{pk_C}(-2) = E_{pk_C}(182 - 24 - 12 - 2) = E_{pk_C}(144)$, which equals $E_{pk_C}(12^2)$.

SecDisCal, as a common protocol in SMC, that has been applied to other related work [118], however, it suffers from inefficiency issue due to its high frequency encryption and communication procedures. Each worker-task pair must be compared to find workers close to a task location. This is computationally expensive and hard to scale to large number of workers and tasks. Therefore, we only use *SecDisCal* to compute distances for promising worker-task pairs. To achieve this goal, the S_L uses a secure indexing (SI) technique in the first stage of HESI, after receiving the encrypted locations of workers, to store the encrypted locations in the SKD-tree structure, which prunes a large number of unnecessary distance computations.

4.6 Secure Indexing

In this section, we present detailed secure indexing techniques. We first introduce the motivation and construction of the SKD-tree, and then discuss how to apply this indexing structure for efficient pruning.

4.6.1 SKD-tree

Recall that it is unnecessary, and time-consuming, to compute distances for each worker-task pair. Given only a small number of workers usually satisfy the neighborhood condition of a task, instead of comparing all worker-task pairs the encrypted locations of workers are indexed in advance, and unpromising workers are pruned before computing the distances.

KD-tree [128] was the first, and most promising, indexing technique we considered to tackle this purpose. A KD-tree is a spatial data structure that organizes points in k-dimensional space. It is a binary tree in which every node is a dimensional point that divides the space into two parts according

to a certain dimension.

Using a KD-tree to construct our framework presents two major challenges. First, all operations must be performed on encrypted data, to ensure that neither the S_L nor the S_C will obtain any private location information during the indexing process. Second, normal KD-trees hold a potential privacy threat. The splitting dimensions of normal KD-trees are predetermined and public – nodes in odd levels split the space with the x-dimension, and nodes in even levels split with y-dimension. Consequently, the S_L could deduce the relative locations of all workers. For instance, it knows w_1 is to the left side of w_2 if w_2 is an x-splitting node and w_1 is in the left subtree of w_2. By continually observing the tree, the possible spatial range of w_1 can be shrunk to a small region, if enough relative location information is collected. Even though the location information is relative, not exact, it is still insecure.

We have therefore developed a novel secure indexing technique based on KD-tree, called SKD-tree. One major difference between an SKD-tree and a normal KD-tree is that SKD-tree is split into two parts and stored on the S_L and S_C separately. The S_L stores the tree structure information (*i.e.*, parent-child relationships) while the S_C stores the dimension splitting information in a dictionary. The splitting dimension of each node is selected randomly, allowing nodes in the same level to split the space along different dimensions. This feature improves security by increasing the difficulty of inferring the relative locations.

Figure 4.3 compares a normal KD-tree *vs.* an SKD-tree using the example in Figure 4.2. The indexing tree is constructed by insertion according to the order from 1 to 7. The graphs show how each node cuts the space according to the splitting dimension. Each node's splitting dimension is predetermined in the normal KD-tree but randomly generated and separately stored on S_C in the SKD-tree. S_L preserves the tree structure and can acquire the dimension information through secure protocols. The shaded nodes in SKD-tree represent data that are encrypted. All worker locations are indexed by a normal KD-tree (left) and an SKD-tree (right). Each line in the graph represents a node that splits the space along a particular dimension. In a normal KD-tree the splitting information is fixed in advance. By con-

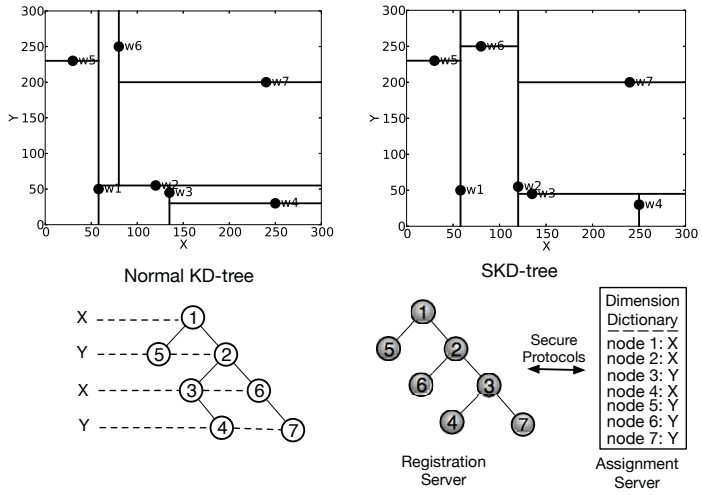


Figure 4.3: An example of a normal KD-tree *vs.* an SKD-tree with reference to worker locations in Figure 4.2.

trast, each node's splitting dimension is randomly generated and separately stored in the SKD-tree on the S_C. The S_L, preserving the tree structure, acquires the dimension splitting information from the S_C through secure protocols (details are given later). The shaded nodes in the SKD-tree represent encrypted data. Both data structures are constructed by inserting nodes one by one from ① to ⑦. Take the insertion of node ⑥ in the SKD-tree for example. Node ⑥ is first compared with the root node ① and assigned to the right sub-tree because ① is X-splitting and ⑥ lies in the right partition of the split along the X axis. Then it is inserted as the left child of ② because node ② is also X-splitting and is in the left part of the partition. We discuss the construction of SKD-tree in detail below.

Before explaining SKD-tree construction details, we introduce an auxiliary protocol, namely *SecVerifySide*, which allows the S_L to find the correct sub-tree for the insertion of a new node. The details of *SecVerifySide* are given in Algorithm 4.3. The input of the protocol from the S_L has two nodes: C is an existing node of the SKD-tree and P is the node to be inserted. The input from the S_C is a dictionary (called *dict*) that contains the splitting information for each node. The splitting dimension is randomly generated on the S_C side by flipping an unbiased coin. The output is a decision as to whether P should be in the left or right sub-tree of C.

Algorithm 4.3 SecVerifySide(P, C)

Input: S_L wants to insert node P to the tree starting from node C. $(E_{pk_C}(l_{P_x}), E_{pk_C}(l_{P_y}))$ and $(E_{pk_C}(l_{C_x}), E_{pk_C}(l_{C_y}))$ are the corresponding encrypted locations. S_C preserves a splitting information dictionary: $dict = \{< w_{id} : dim > | dim \in \{x, y\}\}$.

Output: **true** if P is on C's left sub-tree, and **false** otherwise.

1. S_L:
(a) Pick two random numbers $r_x, r_y \in \mathbb{Z}_N$.
(b) $p'_x = E_{pk_C}(l_{P_x}) * E_{pk_C}(r_x), p'_y = E_{pk_C}(l_{P_y}) * E_{pk_C}(r_y)$,
$c'_x = E_{pk_C}(l_{C_x}) * E_{pk_C}(r_x), c'_y = E_{pk_C}(l_{C_y}) * E_{pk_C}(r_y)$.
(c) Flip a coin c
if c is up **then**
Send ordered tuple $\{h'_x, t'_x, h'_y, t'_y\} = \{p'_x, c'_x, p'_y, c'_y\}$.
else
Send ordered tuple $\{h'_x, t'_x, h'_y, t'_y\} = \{c'_x, p'_x, c'_y, p'_y\}$.
(d) Send $E_{pk_C}(id_C)$ to S_C
2. S_C:
(a) Decrypt and get id_C.
if $dict[id_C]$=x **then**
Compute $\Delta = D_{sk_C}(h'_x) - D_{sk_C}(t'_x)$.
else
Compute $\Delta = D_{sk_C}(h'_y) - D_{sk_C}(t'_y)$.
(b) Send $q' = E_{pk_L}(1)$ to S_L if $\Delta < 0$ and $q' = E_{pk_L}(0)$ otherwise.
3. S_L:
(a) Decrypt $q = D_{sk_L}(q')$.
if c is up **then**
return true if $q = 1$, and **false** otherwise.
else
return true if $q = 0$, and **false** otherwise.

In step 1, S_L injects disturbances by adding random integers to protect the encrypted locations. Since the S_L does not know whether C is x-splitting or y-splitting, it sends both dimensional encryptions to S_C. In order to prevent S_C from deducing any useful information, these ciphertexts are transmitted via an oblivious transfer technique [129]. Specifically, the S_C has two alternative orders of grouping these encryptions and decides the order by flipping an unbiased coin. In this way, the S_C is not able to determine which ciphertext corresponds to which node. In step 2, the S_C looks up the dimension from *dict* and computes the coordinate distance accordingly. Then S_C verifies the subtraction and sends the result to the S_L. The S_L then evaluates the result according to the previous flipped coin information. It can be seen that during the whole protocol, the S_C only does the decryption and subtraction, but does not know which ciphertext represents what identity. Therefore, the S_C is not able to deduce any useful information, even though it knows the distance between P and C.

Based on *SecVerifySide*, we now describe the protocol for inserting a node into the SKD-tree. Given an SKD-tree $\mathcal{S}$ and a new node N, the goal of the *SecInsert* protocol is to insert N into $\mathcal{S}$ securely. The details are given in Algorithm 4.4. The C node is initialized as $\mathcal{S}$.Root (*i.e.*, the algorithm is called as *SecInsert*($\mathcal{S}$.Root, N, $\mathcal{S}$)). The algorithm compares N with the nodes in the tree iteratively by calling *SecVerifySide*. When it reaches a leaf node, it inserts N under the leaf node according to the verification result.

Note that in our current implementation, the S_L constructs an SKD-tree by using a randomly selected worker as the root node and inserting the remaining workers one by one. Though the SKD-tree may be imbalanced, our experiments show that the constructed tree enables effective pruning and the randomized construction has good scalability.

4.6.2 Fast Pruning

A constructed SKD-tree can be used to efficiently find a set of nodes (*e.g.*, workers) within a spatial range (*e.g.*, close to a task). Because the S_L only stores the tree structure, not the splitting information, we have devised an interactive protocol to complete the pruning securely.

Algorithm 4.4 SecInsert(C, N, $\mathcal{S}$)

Input: S_L has an SKD-tree S. S_C has *dict* and *sk*. C is the current node in the iteration.
Output: N is inserted to $\mathcal{S}$.

```
while C is not a leaf node do
  if SecVerifySide(N, C) then
    //N is on C's left sub-tree
    C=C.LeftChild
  else
    //N is on C's right sub-tree
    C=C.RightChild
if SecVerifySide(N, C) then
  Insert N as C's left child.
else
  Insert N as C's right child.
```

To prune unpromising workers, we conservatively use the maximum value of workers' travel distances as the search range, *i.e.*, $\hat{D} = \max_{w_i \in \mathcal{W}} D_{w_i}$. Given a task $t \in \mathcal{T}$ and $\hat{D}$, we can securely compute a *range search rectangle*, denoted by *Rect*, using homomorphic encryption. More precisely, *Rect* is a tuple of four encrypted vertexes (x^+, x^-, y^+, y^-), where

$$\begin{aligned} x^+ &= E_{pk_C}(l_{t_x}) * E_{pk_C}(\hat{D}), \\ x^- &= E_{pk_C}(l_{t_x}) * E_{pk_C}(\hat{D})^{N-1}, \\ y^+ &= E_{pk_C}(l_{t_y}) * E_{pk_C}(\hat{D}), \\ y^- &= E_{pk_C}(l_{t_y}) * E_{pk_C}(\hat{D})^{N-1}. \end{aligned}$$

The left-bottom and right-upper vertexes are (x^-, y^-) and (x^+, y^+), respectively, which are computed on the S_L. To make use of the rectangle range, we can prune the unpromising workers by simply comparing the target's x- and y-coordinates with *Rect* accordingly. Our aim is to find a set of workers whose locations are within *Rect*.

In the following, we first propose an auxiliary protocol *SecComp* to securely compare two encrypted numbers. The protocol is demonstrated in Algorithm 4.5. Given two encrypted inputs, the S_L sends the perturbed inputs to the S_C, which decrypts the received ciphertexts, calculates the sub-

traction, and notifies the S_L of the comparison results. Then, the S_L evaluates the result based on the flipped coin. In the SC scenario, this protocol is used to verify the coordinate differences between two encrypted locations.

Algorithm 4.5 SecComp$(E_{pk_C}(a), E_{pk_C}(b))$

Input: S_L has $E_{pk_C}(a)$, $E_{pk_C}(b)$ and S_C has sk.
Output: **true** if $a < b$, and **false** otherwise.
1. S_L:
(a) Pick a random number $r \in \mathbb{Z}_N$.
(b) $a' = E_{pk_C}(a) * E_{pk_C}(r)$, $b' = E_{pk_C}(b) * E_{pk_C}(r)$
(c)Flip a coin c
if c is up **then**
Send ordered tuple $\{h', t'\} = \{a', b'\}$ to S_C.
else
Send ordered tuple $\{h', t'\} = \{b', a'\}$ to S_C.
2. S_C:
(a) Compute $\delta = D_{sk_C}(h') - D_{sk_C}(t')$.
(b) Send $q' = E_{pk_L}(1)$ to S_L if $\delta < 0$ and $q' = E_{pk_L}(0)$ otherwise.
3. S_L:
(a) Decrypt $q = D_{sk_L}(q')$.
if c is up **then**
return **true** if $q = 1$, and **false** otherwise.
else
return **true** if $q = 0$, and **false** otherwise.

Based on the above protocol, we use *SecSearch* for the range search in SKD-tree. As shown in Algorithm 4.6, the whole search protocol is an iterative process. In each iteration, the algorithm verifies the spatial relationship between the current node C and the range rectangle *Rect*. If C lies on the left or at the bottom of *Rect*, the S_L turns to C's right child for further comparison. Similarly, if C is to the right or on top of *Rect*, C's left child will be compared. If C is within *Rect*, we add C to the result set and continue to inspect both of C's children. All the comparisons are completed by evaluating C's coordinates with *Rect*'s boundaries using *SecComp*. The following lemma ensures the effectiveness of *SecSearch*.

Lemma 4.1. SecSearch *is correct and complete.*

Proof. To prove the correctness, we need to verify that all workers in the set of discovered $\mathcal{W}_r$ are inside *Rect*. This can be straightforwardly achieved

Algorithm 4.6 SecSearch(C, *Rect*)

Input: S_L has the SKD-tree $\mathcal{S}$ and S_C has *Dict* and *sk*. C is a node in $\mathcal{S}$ and *Rect* is the required query rectangle.

Output: S_L obtains a worker set $\mathcal{W}_r$ whose real locations are within the specified *Rect*.

S_L:

Send C to S_C and execute the following protocols simultaneously: *SecComp*($E_{pk_C}(l_{C_x})$, x^-), *SecComp*($E_{pk_C}(l_{C_x})$, x^+), *SecComp*($E_{pk_C}(l_{C_y})$, y^-) and *SecComp*($E_{pk_C}(l_{C_y})$, y^+).

S_L and S_C:

Lookup $dim = dict[id_C]$.

if $dim = x$ **then**

  if *SecComp*($E_{pk_C}(l_{C_x})$, x^-) **then**

    //The target is on C's right sub-tree

    Let S_L invoke *SecSearch*($C.RightChild$, *Rect*).

  if not *SecComp*($E_{pk_C}(l_{C_x})$, x^+) **then**

    //The target is on C's left sub-tree

    Let S_L invoke *SecSearch*($C.LeftChild$, *Rect*).

  else

    //C is the potential target.

    if *SecComp*($E_{pk_C}(l_{C_y})$, y^+) **and**

    not *SecComp*($E_{pk_C}(l_{C_y})$, y^-) **then**

      //C is the desired node

      Add C to $\mathcal{W}_r$.

    SecSearch(C.LeftChild,*Rect*)

    SecSearch(C.RightChild,*Rect*)

else

  (same as *then* clause with "x" and "y" exchanged)

because, according to the protocol execution, $\forall w \in W_r$, $x^- \leq l_{w_x} \leq x^+$ and $y^- \leq l_{w_y} \leq y^+$ must be satisfied simultaneously.

The completeness is proved as follows. Let the true result set denoted by $\mathcal{W}_r^*$, the set of pruned nodes be $\bar{\mathcal{P}}$ and the set of other nodes be $\mathcal{P}$. Note that $P \cap \bar{\mathcal{P}} = \emptyset$ and $\mathcal{P} \cup \bar{\mathcal{P}} = \mathcal{W}$. Because the algorithm will examine all the nodes in $\mathcal{P}$, the aim is to prove for each iteration $\forall w \in \mathcal{W}_r^*, w \notin \bar{P}$.

There are three possible results in each round when C is compared with *Rect*: (1) C is to the left (or on the top) of *Rect*, (2) C is to the right (at the bottom) of *Rect*, or (3) otherwise. In the first two situations, we can safely prune a whole branch of the subtree because it can be assured that the nodes

in $\bar{\mathcal{P}}$ are not inside *Rect*. In the third situation, the algorithm does not prune any nodes. Instead, it adds the nodes of both subtrees into $\mathcal{P}$. Therefore, for all iterations, the nodes added to $\bar{\mathcal{P}}$ will not appear in the result set. Combined with the correctness, completeness is obtained. □

Referring to Figure 4.2, for example, suppose we want to search the workers in the vicinity of t_2 within the range of 50 (*i.e.*, $\hat{D} = 50$). Since $l_{t_2} = (140, 90)$, we have $x^+ = E_{pk_C}(190)$, $x^- = E_{pk_C}(90)$, $y^+ = E_{pk_C}(140)$ and $y^- = E_{pk_C}(40)$. The search process starts from w_1 (the root of the tree), which is an X-splitting node according to the dictionary inquiry. Because $SecComp(l_{w_{1_x}}, x^-) = smaller$, it means that *Rect* does not include w_1 and it is on w_1's right hand side. Thus it is not necessary to examine the left child (w_5) and the algorithm turns to the right sub-tree. In the second iteration, w_2 is verified as being inside *Rect*, hence the algorithm adds w_2 to the result set $\mathcal{W}_r$, and continues the search by exploring both of w_2's sub-trees. Similarly, after the comparison, we can prune the left child w_6 and add w_3 to the result set. In the final round, we examine both of w_3's sub-trees and filter out w_4 and w_7. Therefore, the final result set is $\mathcal{W}_r = \{w_2, w_3\}$.

In practice, the size of $\mathcal{W}_r$ will be small as there are only limited number of workers close to a task, which means generally that a great number of nodes can be pruned during each iteration. Thus *SecSeach* can output the desired worker set efficiently, for which we compute the exact distance between task and worker using *SecDisCal*. Our experiments in Section 4.9 also show that the proposed protocol is scalable for large datasets.

4.7 Secure Task Assignment

In this section, we first study the task assignment strategy, and then propose a secure assignment framework.

4.7.1 Assignment Strategy

Many existing task assignment models, such as [75], [116] and [130], concern the criterion of minimizing overall worker travel costs and can be applied to our framework. Note that our main contribution is to protect lo-

cation privacy. To make the process of spatial crowdsourcing complete, we borrow the assignment strategy from [116] and introduce our assignment strategy as follows.

Suppose that not all workers are trustworthy and there is an associated probability value concerning the likelihood of certain tasks being performed, thus the task acceptance is probabilistic. Let the probability of a worker performing a task (called *acceptance probability*) depend on the distance between their location and the target's location. More precisely, given a task t_i and a worker w_j, we assume the acceptance probability decreases linearly as the distance becomes larger, and is zero when the distance exceeds the worker's maximum travel distance. The acceptance probability can be formally defined as

$$AP_{t_i w_j} = \begin{cases} -\frac{||l_{t_i} - l_{w_j}||}{D_{w_j}} + 1, & \text{if } 0 < ||l_{t_i} - l_{w_j}|| < D_{w_j} \\ 0, & \text{otherwise} \end{cases}$$

Based on the acceptance probability of workers with respect to a task, a *confidence probability* can be defined for each task indicating how likely the task will be accepted. A task is successfully assigned if the confidence probability is greater than a given threshold (*i.e.*, denoted as α). Given a set of k workers and a task t_i, the confidence probability is defined as $CP_{t_i} = 1 - \prod_{j=1}^{k}(1 - AP_{t_i w_j})$, which is the probability that at least one worker will accept the task. If the confidence probability is no less than α, the task can be assigned to these k workers. In fact, there are multiple groups of workers available for a task.

Another restriction of the task assignment is the acceptance number T_{w_j} for each worker, which prohibits simultaneously assigning more than T_{w_j} tasks to a worker w_j.

Overall, given the task set $\mathcal{T}$, the worker set $\mathcal{W}$, the distance matrix $\mathcal{M}$, and the confidence probability threshold α, task assignment aims to successfully assign as many tasks as possible, under the conditions:

$$\forall t_i \in \mathcal{T}, w_j \in \mathcal{W},\ ||l_{t_i} - l_{w_j}|| \leq D_{w_j},$$
$$CP_{t_i} \geq \alpha, \text{ and } N_{w_j} \leq T_{w_j}.$$

This problem is similar to the one proposed in [116], which has been proved to be an NP hard problem. Below we introduce a greedy approach. For each task, we select one proper worker set as the result from the potential assignment sets if this selected set (1) has the shortest average distance; and (2) does not contradict the restriction. The selection step stops when no more proper assignments exist. For illustration, we use example in Figure 4.2 to demonstrate the task assignment. Suppose maximum travel distance of all the workers is ($\hat{D}$) is 100. After the pruning process, the distance and the acceptance probability matrices are given as:

$$\begin{pmatrix} & w_1 & w_2 & w_3 \\ t_1 & 19.2 & 51.0 & 68.0 \\ t_2 & 91.2 & 40.3 & 45.3 \\ t_3 & 23.9 & 79.1 & 91.2 \end{pmatrix}$$

We can derive the acceptance probabilities as follows:

$$\begin{pmatrix} & w_1 & w_2 & w_3 \\ t_1 & 0.81 & 0.49 & 0.32 \\ t_2 & 0.09 & 0.60 & 0.55 \\ t_3 & 0.76 & 0.21 & 0.09 \end{pmatrix}$$

Let $\alpha = 0.7$, we can group workers and produce the potential worker groups as shown in Table 4.2.

t_1	$w_1, < w_1, w_2 >, < w_1, w_3 >, < w_1, w_2, w_3 >$
t_2	$< w_2, w_3 >, < w_1, w_2, w_3 >$
t_3	$w_1, < w_1, w_2 >, < w_1, w_3 >, < w_1, w_2, w_3 >$

Table 4.2: Potential assignments for each task.

Suppose $T_{w_1} = 2$, $T_{w_2} = 1$ and $T_{w_3} = 1$. Firstly, for task t_1, w_1 is the nearest candidate, thus t_1 is assigned to w_1. Next, because the set $< w_2, w_3 >$ has a shorter average distance (42.8) and does not violate the restriction, t_2 is assigned to this set. Lastly, since w_1 is able to accept one more task and it is closest to t_3, we assigned t_3 to w_1.

4.7.2 Secure Assignment

To adopt the assignment strategy to our framework, we need to compute worker's acceptance probability based on the task-worker distance. One challenge is that if we adopt *SecDisCal* directly to allow S_C to obtain the distance, it may induce a security threat. In particular, if S_C is corrupted and observes a set of tasks and the corresponding encrypted values in advance (also known as the known-ciphertext attack [123]), some distance-recovery attack [108] can be carried out to deduce other workers' locations. For example, if three tasks are leaked, the corrupted S_C may get the distances between w_i and these tasks, establish three equations and consequently deduce the real location of w_i by solving these equations.

A necessary condition to breach the location privacy by leveraging the Euclidean distance results is that the adversary establishes equations by using tasks real locations in advance. To prevent the adversary from establishing such equations, we can hide the task and worker identifies and break their relationships. Following this intuition, we propose to carry out a random permutation on the workers' identities for each task. In this case, though the S_C knows the distance values, it cannot successfully deduce the real identities of workers. As a result, a secure protocol, *SecAssign*, is proposed in Algorithm 4.7, in which both servers carry out the task assignment interactively.

Assume that unnecessary task-worker pairs are already pruned by running *SecSearch* and let $\mathcal{W}_{r_i}$ be the resulting worker sets for the task t_i after pruning. For each task, a random permutation is applied to the neighboring workers' identities. The objective of the permutation is to muddle the identities of the task-worker pairs so that S_C is not able to know which distance value belongs to which pair. Note that the random permutation is independently chosen for each task. Only the S_L possesses the random permutations and can recover the real mapping. In the following, the '*' symbol corresponds to the disturbed identities. After executing *SecDisCal*, the S_L sends the encrypted distance to the S_C. Afterwards, the S_C packs the received distances into a distance vector D_i^*, and decrypts the encrypted distances. Then the potential worker groups are derived based on the confidence probabilities and other assignment restrictions. Note that for the S_C

Algorithm 4.7 SecAssign

Input: S_L has the promising worker set $\mathcal{W}_{r_i}$ for each task
Output: S_L knows the task assignment results
for $i = 1$ to $|\mathcal{T}|$ **do**
Permute workers' identities $\mathcal{W}^*_{r_i} = \Pi_i(\mathcal{W}_{r_i})$
for $w^*_j \in \mathcal{W}^*_{r_i}$ **do**
S_L and S_C: Execute $SecDisCal(E_{pk_C}(l_{t_i}), E_{pk_C}(l_{w^*_j}))$
S_L: Send $d^*_j = E_{pk_C}(|l_{t_i} - l_{w_j}|^2)$ to S_C
S_C:
(a) Pack the distances d^*_j into a distance vector D^*_i
(b) Decrypt the distance vector $M^*_i = D_{sk_C}(D^*_i)$
(c) Calculate the acceptance probability vector P^*_i based on M^*_i
(d) Derive the potential worker groups G^*_i based on P^*_i and other restrictions
(e) Send $C^* = E_{pk_L}(G^*_i)$ to S_L
S_L:
(a) Decrypt $G^*_i = D_{sk_L}(C^*)$
(b) Restore $G_i = \Pi_i^{-1}(G^*_i)$
S_L:
Carry out the assignment based on the potential worker groups using greedy approach

the identities of the workers are disturbed, therefore the real task-worker relationships are unclear for the S_C. The resulting worker groups are then encrypted using pk_L and sent to the S_L. Next, the S_L performs the decryption and re-permutation to obtain the real potential worker groups. Finally, the S_L carries out the assignment based on all the potential worker groups using the greedy approach discussed in the above subsection.

4.8 Analysis

In this section we analyze the security and complexity of the proposed framework.

4.8.1 Security Analysis

In this subsection, we analyze the security of SKD-tree and the proposed protocols accordingly.

The related security analysis of *SecDisCal* and *SecMul* can be found in [118]. In the following, we focus on the secure index protocols and secure assignment protocols.

SecVerifySide: We first analyze the location privacy information the S_C can observe during the execution. The S_C only knows the identity of P, and the partial coordinate differences between the inserted node C and P. Therefore, it cannot learn any sensitive information. Moreover, the S_C does not know whether C is inserted as P's left or right sub-tree. The probability of guess is $1/2$. However, since the coin flipping is independent each time, the probability of deriving the inserting path of C is quite small.

SecComp: Similar to the analysis of *SecVerifySide*, the S_C is not able to know which number is larger, while the S_L can obtain the final result.

SecInsert & *SecSearch*: These two protocols mainly use *SecVerifySide* and *SecSeach* as sub-routines, therefore the security is implied.

SecAssign: The random permutation of workers' identities for each task makes it difficult for the adversary to derive the workers' real locations. After the permutation, although the distance value is disclosed, the identity of the task-worker pair corresponding to the distance value remains unknown. In this case, an adversary is not able to correctly establish the equations. In addition, the adversary is not able to leverage these equations to derive the distance, because the relations between these equations are confused. As a result, a distance recovery attack (such as the one in [108]) cannot be successful.

To formally prove the security of the proposed protocols, we adopt a simulation-based proof technique. In this technique, there two different worlds: the 'real' world and the 'ideal' world. The real world refers to the scenarios where the actual protocol executes. The ideal world exists as an external trusted party that can help all other parties do the computation. It is assumed that no attacks can be carried out in the ideal world. Therefore, if we can construct a simulator to show that the real protocol behaves like an idealized one, the security is implied. The simulator typically works by

simulating a corrupted party in the ideal world. The only thing the simulator can do in the ideal world is to choose the corrupted party's input.

The formal security definitions of a protocol in the semi-honest model in SMC are as follows.

Definition 4.1. *Let $\{D_n\}_{n\in\mathbb{N}}$ and $\{E_n\}_{n\in\mathbb{N}}$ be two distributions or ensembles indexed by a security parameter n, then we say they are **computationally indistinguishable**, if for any non-uniform polynomial time algorithm A the following quantity is a negligible function in n:*

$$\delta(n) = |\Pr_{x\in D_n}[A(x) = 1] - \Pr_{x\in E_n}[A(x) = 1]|.$$

In other words, any polynomial-time algorithm trying to distinguish between these two distributions will only perform negligibly better than if one were to just guess.

Definition 4.2. *[119] Let P be a protocol for computing a function f by two parties. Suppose party A (resp. B) computes the function $f_A(x, y)$ (resp. $f_B(x, y)$), where x, y are the inputs of A and B, respectively. The view of a party is what it sees during the execution. The view of A (resp. B), denoted by V_A (resp. V_B), is defined as:*

$$V_A = (x, r, m_1, \ldots, m_t)$$
$$V_B = (y, r, m_1, \ldots, m_t)$$

where r represents the randomness and m_i represents the messages passed between parties.

We say protocol P is secure in the semi-honest model if there exists probabilistic polynomial time (PPT) simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ such that:

$$\mathcal{S}_1(x, f_A) \equiv V_A \tag{4.3}$$
$$\mathcal{S}_2(y, f_A) \equiv V_B \tag{4.4}$$

where $\equiv$ denotes computational indistinguishability.

In other words, a protocol is secure if its execution can be simulated such that the distribution of the simulated execution is computationally indistinguishable from the real one.

In our work, all proposed protocols are running by two parities. We summarize the characteristics of the proposed protocols and prove their security by the following theorem.

Theorem 4.1. *Suppose that A and B run a protocol P, in which all messages passed from A to B are encrypted using a secure homomorphic encryption scheme, and all messages passed from B to A are uniformly distributed and are independent of B's inputs. Then we say protocol P is secure under the semi-honest adversaries model.*

Proof. Firstly, consider the scenario that A is corrupted. In this case, we can construct a simulator, $\mathcal{S}_1$, to simulate A's view in the following way: for every message that A receives from B, $\mathcal{S}_1$ randomly picks a random number from a certain domain and encrypts it. Since the encryption scheme is a semantically secure scheme that generates ciphertexts which are uniformly distributed, the simulated message is computationally indistinguishable from the real one. That is, equation (4.3) holds. Secondly, if B is corrupted, we construct a simulator $\mathcal{S}_2$ to simulate the messages sent by A. Similarly, let $\mathcal{S}_2$ picks a random element and encrypts it. Any PPT adversary cannot distinguish the simulators encryption of a random number from A's encryption of the correct computation, which means equation (4.4) holds. □

By this theorem, the protocols *SecDisCal*, *SecMul*, *SecVerifySide*, *SecComp* and *SecAssign* are secure, because the messages transmitted in these protocols are either ciphertexts or random numbers that are independent of the private input. The security of *SecInsert* and *SecSeach* are also ensured because they are the sub-routines of *SecVerifySide* and *SecComp*, respectively.

4.8.2 Complexity Analysis

We analyze the complexity of HESI by the number of operations, *i.e.*, encryption, decryption and exponentiation. We assume that all operations using the Paillier cryptosystem cost a similar amount of time. Suppose there are m tasks and n workers in total.

When inserting a node in the SKD-tree, it costs 4 encryptions and 3 decryptions to find the appropriate position by calling *SecVerifySide*, *i.e.*, 7 op-

erations in total. Similar to the binary search tree, the complexity of the SKD-tree for insertion is $O(n)$ in the worst case and $O(\log n)$ on average. Thus, the complexity of constructing an SKD-tree is bounded by $O(n^2)$ at worst and $O(n \log n)$ on average.

The pruning process in the SKD-tree requires 4 encryptions to form the search range *Rect*, and the *SecComp* protocol requires 5 operations (2 encryptions and 3 decryptions). The pruning process is basically a range search problem on a KD-Tree, which has been well studied with the conclusion that the average complexity for a range search in a k-dimensional KD-Tree is $O(k + \log n)$, and for the worst case it can be given by $O(kn^{1-\frac{1}{k}})$ [131]. Therefore in a 2-dimensional space, the complexity of pruning is bounded by $O(\sqrt{n})$ in the worst case and $O(\log n)$ on average.

The distance computation requires 8 cryptographic operations to multiply two integers using *SecMul*. Thus, the complexity of executing *SecDisCal* is $2 + 8 * 2 + 1 = 19$. The total complexity of the distance computation is determined by the number of worker-task pairs for comparison. Basically, the baseline (index-free) solution needs to compare all possible worker-task combinations, which is subject to complexity of $19mn$, which is of order $O(mn)$. In our framework, let q be the maximum number of workers adjacent to a task: the total complexity of computing distances for m tasks is bounded by $19mq$, which is of order $O(mq)$.

In terms of task assignment, let Q be the average complexity for picking a satisfactory worker set from the potential assignment sets for one task. The total assignment complexity is bounded by mQ.

Table 4.3: Complexity summary.

Operations	Index-based		Index-free
	Average	Worst	
Tree construction	$O(n \log n)$	$O(n^2)$	-
Pruning	$O(m \log n)$	$O(m\sqrt{n})$	
Distance computation	$O(mq)$		$O(mn)$
Task assignment	mQ		

Table 4.3 summarizes the details. By comparing the computational complexity, with the exception of the task assignment step, it can be seen that our index-based approach is bounded by $O((m + n) \log n + mq)$ on average

while the index-free method costs $O(mn)$. The advantage of indexing will be more significant when n is large. Our experiment results show that the index-based approach has good scalability and significantly improves the performance.

For each worker or task requester, the computational cost at the client side (excluding server computational costs) is comparable to non-encryption approaches, because the client side only needs to perform a small number of encryption/decryption operations, which costs less than one second according to our experiments.

4.9 Performance Evaluation

In this section, we evaluate the performance of the proposed framework using two real-world datasets. All experiments were implemented in C++ on a Linux machine with a 3.47GHz Intel Xeon CPU and 12GB RAM.

4.9.1 Benchmark Data

The **Yelp dataset** is a collection of user reviews about local businesses, such as restaurants. It includes users' comments, check-ins and business information. We consider each Yelp user as an SC worker with their check-in as the location, and assume that the restaurants are the specified task targets. The **Gowalla dataset** is a location-based social network dataset where users share their locations with their friends. Each Gowalla user is considered to be an SC worker, and their location is the most recent check-in. Each check-in point is also modeled as a task location.

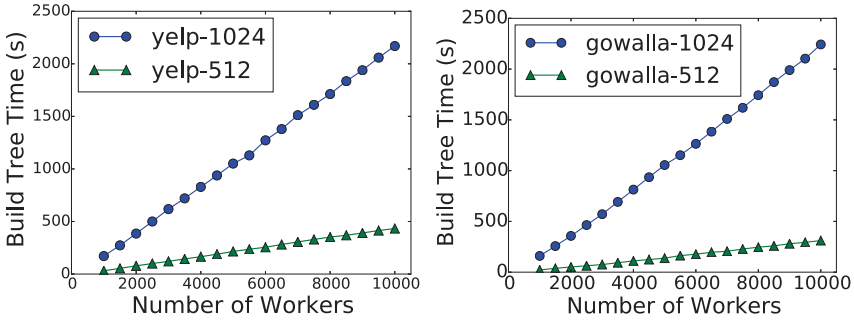
In our experiments, $10,000$ workers and $5,000$ tasks were chosen from both datasets. It is assumed that each worker's maximum number of tasks (T_i) and maximum travel distance (D_i) are the same. By default, we set T_i to 5, set $D_i = 1km$ for the Yelp dataset and $D_i = 10km$ for the Gowalla dataset. All geographic coordinates of locations are transformed to projection coordinates using the Universal Transverse Mercator (UTM) projection. We ran each experiment five times and report the average runtime.

4.9.2 Experimental Results

4.9.2.1 SKD-tree Evaluation

In this subsection, we evaluate the performance of SKD-tree from different aspects.

We first evaluate the scalability of our tree construction method. Two Paillier key sizes are used: $K = 512$ bits and $K = 1024$ bits. We vary the number of workers from 1000 to $10,000$ and record the corresponding runtime for building the SKD-tree. The results are presented in Figure 4.4 and show that the runtime for tree construction achieves good scalability on both datasets. In addition, we observe that the encryption key size influences the performance significantly, which justifies the fact that a trade-off between privacy and efficiency exists. For example, when the number of workers in the Yelp dataset is $10,000$, the tree building time is $433s$ for $K = 512$ but increases to $2196s$ when $K = 1024$. However, the tree construction time for $K = 1024$ is still efficient. In the following experiments, we used 1024 as the default key size.



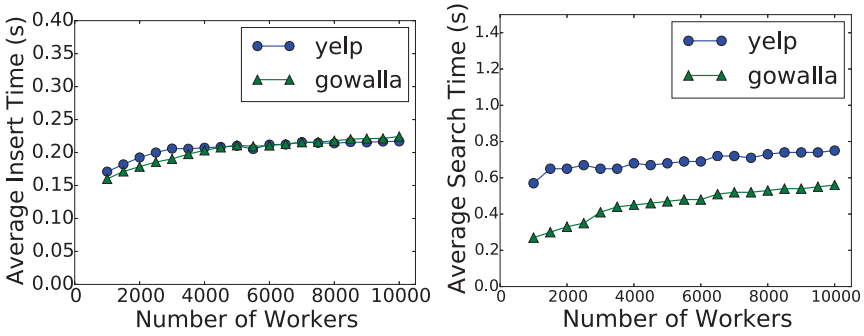
4.4.a Build Tree - Yelp

4.4.b Build Tree - Gowalla

Figure 4.4: Evaluation of tree construction.

Next we evaluate the operations on SKD-tree. Because deletion is very similar to insertion, only the results of insertion and range search are presented. Figure 4.5.a illustrates the costs of inserting a node into trees of different size. The y-axis represents the average runtime of inserting a node (*i.e.* worker) into the tree. It can be observed that the insertion time increases on both datasets as the tree size increases. The reason is that more comparisons are needed to find an appropriate position for insertion when the tree

becomes larger. However, it can be observed that the trend increases quite slowly, showing that the insertion operation is scalable to the tree size.



4.5.a Insert Time vs. Tree Size 4.5.b Search Time vs. Tree Size

Figure 4.5: Tree operation evaluation.

To evaluate range search, we invoked 100 random queries with a default range size of $1km$. Figure 4.5.b reports the average search time with respect to the number of workers, which demonstrates good scalability on both datasets. Moreover, it costs less time to search the Gowalla dataset because it is sparser than the Yelp dataset, more unnecessary comparisons are pruned at each step.

In conclusion, our proposed SKD-tree is scalable in construction, insertion and search, and can be applied to large scale SC environments.

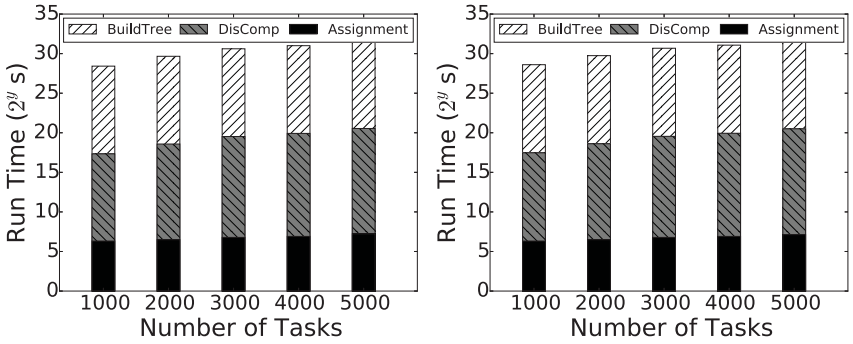
4.9.2.2 Overall Performance Evaluation

We compared the running time of building a tree, computing distances and assigning tasks by varying the number of workers and tasks on both datasets respectively. Figure 4.6 reports the results by plotting the logarithmic runtime of corresponding components. It can be observed that the distance computation accounts for the major time consumption, and the costs of task assignment are comparatively low.

Figures 4.6.a and 4.6.b show that the runtime of distance computation increases linearly with respect to the number of tasks. Taking the Yelp dataset as an example, the runtime increases from $20.7s$ to $43.6s$ when the task number changes from 1000 to 2000. It demonstrates that our proposed protocols

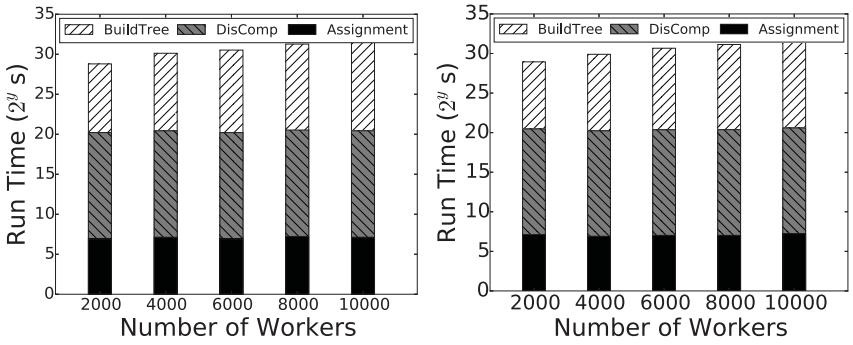
of computing distance have good scalability with respect to the number of tasks.

Conversely, Figures 4.6.c and 4.6.d show that the number of workers hardly has any effect on the runtime of distance computation. This is because the complexity of distance computation does not increase according to the actual number of workers. Many unnecessary distance computations can be discarded by effective pruning enabled by the secure indexing technique.



4.6.a Yelp w.r.t. tasks

4.6.b Gowalla w.r.t. tasks



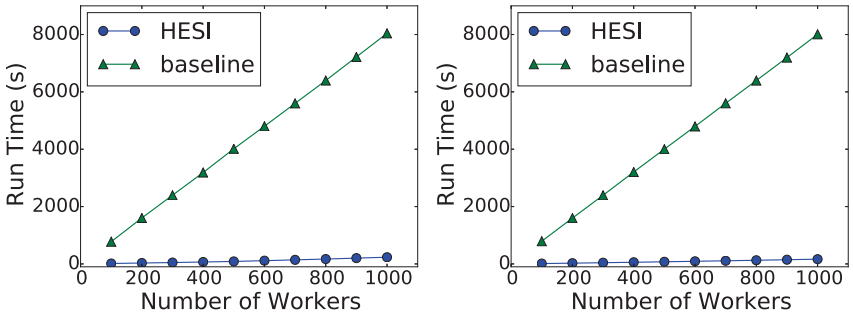
4.6.c Yelp w.r.t. workers

4.6.d Gowalla w.r.t. workers

Figure 4.6: Overall Performance.

Secondly, we compare the performance of HESI to the baseline index-free framework which compares all worker-task pairs for distance information. We set the number of tasks as 10 and varied the number of workers from 100 to 1000. The results are shown in Figure 4.7. It can be seen that the performance of the baseline framework increases linearly with respect to the number of workers. The HESI framework shows a clear advantage over the baseline. This is mainly because HESI is able to prune a large number

of unnecessary distance computations with the contribution of the secure indexing technique.



4.7.a Yelp 4.7.b Gowalla

Figure 4.7: Performance Improvement.

Lastly, we evaluate the influence of location distribution on distance computation using synthetic data. We generate three types of location distribution: *uniform*, *clustered* and *skewed* distribution for worker and task locations. A *uniform* distribution was generated by randomly picking 1000 points from a $100km \times 100km$ rectangle centered on Los Angeles. The *clustered* distribution consists of 10 clusters, each containing 100 points and having an isotropic Gaussian distribution with a variance of $20km$. The *skewed* distribution involves only one big cluster with 1000 points. We recorded the runtime of distance computation for different worker and task location distribution. Table 4.4 reports the results.

		Tasks		
		Uniform	Clustered	Skewed
Workers	Uniform	0.28s	206s	218s
	Clustered	0.24s	252s	228s
	Skewed	0.16s	143s	85s

Table 4.4: Performance of distance computation for different location distribution.

It can be seen that the distance computation runs longer when the distribution of task locations is clustered or skewed, regardless of the variation in the distribution of worker locations. This can be explained as follows. Recall that we invoke *SecSearch* protocol to find nearby workers for each

task. The cost of the search process can be neglected for those tasks without nearby workers, because it is only necessary to compare the root node of SKD-tree when executing the protocol. For comparison, we consider the queries for tasks involving nearby workers as *costly queries*, and assume the corresponding costs are the same. Therefore, the results in Table 4.4 can be explained by the fact that when the distribution of task locations is clustered or skewed, the number of costly queries is large, resulting in longer running time. For example, let us consider a simple clustered situation with 1000 task-worker neighboring pairs in total. If there are 100 clustered tasks and 10 workers located near the cluster center, there are 100 costly queries. By contrast, if there are 100 workers within a cluster and 10 tasks are closed to the cluster center, only 10 costly queries are needed.

4.9.2.3 Communication Cost Evaluation

In this section, we evaluate the communication overhead between two servers in the proposed framework. Specifically, we record the total size of data that transferred during the executions of the secure protocols. Table 4.5 shows the results with respect to different numbers of workers and tasks. It can be seen that the cost changes from 8.18MB to 26.25MB when the number of workers varies from 2000 to 10000, and changes from 5.24MB to 21.83MB when the number of tasks varies from 1000 to 5000. The result shows that the extra communication overhead due to the dual-server design is acceptable, and our proposed framework is feasible in practice.

Table 4.5: Communication cost.

#Workers	Comm. Cost (MB)	#Tasks	Comm. Cost (MB)
2000	8.18	1000	5.24
4000	10.95	2000	8.63
6000	16.20	3000	13.35
8000	20.22	4000	17.39
10000	26.25	5000	21.83

4.9.2.4 Task Assignment Evaluation

For comparison, a baseline assignment model is simulated by assigning a task to a nearby worker who decides whether to accept the task request based on the distance. In this experiment there are 1000 workers and 5000 tasks. Each worker's T_i equals 5 and $\hat{D}$ is chosen from [1km, 5km, 10km] for Yelp and [10km, 15km, 20km] for Gowalla. We examine the number of assigned tasks and the average travel cost. Note that a worker may accept multiple tasks. In this case the cost is computed by averaging the distances with respect to the number of the tasks he/she accepted. The results are presented in Table 4.6 and Table 4.7, respectively. It can be seen that our assignment solution is able to disseminate more tasks to workers while maintaining lower travel costs. Hence, our proposed solution is effective for SC task assignment.

	$\hat{D} = 1km$		$\hat{D} = 5km$		$\hat{D} = 10km$	
	base	HESI	base	HESI	base	HESI
NO. of tasks	539	743	593	834	641	898
Avg. cost (km)	0.81	0.72	4.02	3.43	6.21	5.87

Table 4.6: Task assignment evaluation on Yelp.

	$\hat{D} = 10km$		$\hat{D} = 15km$		$\hat{D} = 20km$	
	base	HESI	base	HESI	base	HESI
NO. of tasks	483	531	553	674	694	752
Avg. cost (km)	6.13	5.47	11.23	9.42	14.61	12.94

Table 4.7: Task assignment evaluation on Gowalla.

4.10 Conclusions

In this chapter, we proposed a novel privacy-preserving framework for spatial crowdsourcing, which ensures that user locations are never released to anyone, yet the system is still able to assign tasks to workers in an efficient way. In particular, we encrypt all location information using the Paillier cryptosystem. To calculate worker-task distance without knowing the

actual locations of workers and tasks, we proposed a dual-server design which uses homomorphic encryption to conduct computations based on encrypted data. To improve the efficiency, we advocated a secure indexing technique and proposed an SKD-tree to index all encrypted worker locations for fast pruning. The key innovation of our framework, compared to existing work in the field, is threefold: (1) a new encrypted data based spatial crowdsourcing framework for the SC community; (2) a secure SKD-tree structure to store and index encrypted data for fast search; and (3) ensured data privacy (including worker and requester privacy) and data security, whereas existing works only limit to worker privacy. In our future work, we will study the optimization of the SKD-tree for further improvement.

Moreover, since our framework provides useful operations such as secure distance computation, it can also be used to support a variety of other secure data mining tasks. For example, it can be adopted in the secure data clustering problem where all data are encrypted, because certain secure computing operation are maintained.

Chapter 5

Conclusion

As the arrival of big data era, spatial data mining is facing new challenges. In this thesis, we have investigated the challenges from three aspects: uncertainty, condensity, and privacy.

Firstly, we discuss the motivation of uncertain data in real-world application. On one hand, the data is inherently uncertain in many applications. On the other hand, artificial noise may be added deliberately for privacy protection. It is difficult to process uncertain data because new algorithms are necessary for the same mining task, and usually they require high computation costs. To show how to solve this challenge, we have studied the problem of discovering co-location patterns in the context of continuously distributed uncertain data. The spatial data are modeled as multivariate Gaussian distributions, based on which we formulate the problem of probabilistic Co-location pattern mining and develop a efficient framework for computing and verifying the neighborhood relationship between instances. Our techniques can be applied to other scenarios where the original data are represented as multivariate Gaussian distributions.

Secondly, we discuss the necessity of summarizing output patterns. A spatial data mining framework may generate a large amount of patterns, or if setting a high threshold, only commonsense patterns are returned. Co-location pattern mining also suffers this problem. We propose an effective way to reduce the resulting pattern numbers, while the outputs still deliver interesting and useful knowledge. Based on the definition of co-location, we define a new measure to appropriately quantify the prevalence distance be-

tween two co-location patterns. Then two efficient algorithms are proposed for summarization. One is a post-mining framework that finds representative patterns from the set of discovered prevalent co-location patterns. The other one, which is more efficient, is to discover representative patterns directly from the spatial data set. Though the techniques are devised for co-location pattern mining, we believe that the ideas can be applied to other pattern mining tasks, especially the mine-and-summarize paradigm.

Thirdly, we address the privacy issue and focus on the problem of protecting location privacy in spatial crowdsourcing. Different from traditional privacy-preserving techniques, we encrypt the target location data using cryptographic algorithms, and make sure the plaintext will not expose to the unwanted parties during the whole time. The challenge lies in how to process encrypted data, *e.g.*, numerical comparing and distance computing, without decrypting the ciphertext. We adopt a duel-server setting, and design security protocols covering the process from worker registration to task assignment in spatial crowdsourcing. To improve the performance, we devise a novel secure indexing technique to index all encrypted data for fast data retrieval. We show that the security and efficiency is a trade-off and it takes efforts to balance these two factors. The proposed framework and techniques are not limited to spatial crowdsoucing scenario. It can be easily extended to other cloud computing applications where user data are encrypted and outsourced.

The challenges of SDM are not limited to these three prospectives. In fact, as the development of big data analysis, traditional mining techniques are not able to handle new characteristics of spatial data, such as fast variation, continual evolving and large volume. For future work, we will investigate these challenges more thoroughly, and apply the resulting results to solve more real-world problems.

References

[1] Shashi Shekhar, Pusheng Zhang, Yan Huang, and Ranga Raju Vatsavai. *Data Mining: Next Generation Challenges and Future Directions*. AAAI/MIT Press, 2003.

[2] Deren Li, Shuliang Wang, Hanning Yuan, and Deyi Li. Software and applications of spatial data mining. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 6(3):84–114, 2016.

[3] Majid Shishehgar, Seyed Nasirodin Mirmohammadi, and Ahmad Reza Ghapanchi. A survey on data mining and knowledge discovery techniques for spatial data. *IJBIS*, 19(2):265–276, 2015.

[4] Shashi Shekhar, Pusheng Zhang, and Yan Huang. Spatial data mining. In *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pages 837–854. 2010.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.

[6] Shashi Shekhar, Paul R. Schrater, Ranga Raju Vatsavai, Weili Wu, and Sanjay Chawla. Spatial contextual classification and prediction models for mining geospatial data. *IEEE Trans. Multimedia*, 4(2):174–188, 2002.

[7] Pusheng Zhang, Yan Huang, Shashi Shekhar, and Vipin Kumar. Exploiting spatial autocorrelation to efficiently process correlation-based similarity queries. In *SSTD*, pages 449–468, 2003.

[8] Baris Kazar, Shashi Shekhar, and David J. Lilja. Parallel formulation of spatial auto-regression. In *Army High-Performance Computing Research Center (AHPCRC) Technical Report no. 2003-125*, 2003.

[9] Monica Wachowicz and Tianyu Liu. Finding spatial outliers in collective mobility patterns coupled with social ties. *International Journal of Geographical Information Science*, 30(9):1806–1831, 2016.

[10] Chongcheng Chen, Jiaxiang Lin, Xiaozhu Wu, and Jianwei Wu. Parallel and distributed spatial outlier mining in grid: Algorithm, design and application. *J. Grid Comput.*, 13(2):139–157, 2015.

[11] S Shekhar and Y Huang. Discovering Spatial Co-location Patterns: a Summary of Results. *Advances in Spatial and Temporal Databases*, pages 236–256, 2001.

[12] Hien Duy Nguyen, Geoffrey J. McLachlan, and Ian A. Wood. Mixtures of spatial spline regressions for clustering and classification. *Computational Statistics & Data Analysis*, 93:76–85, 2016.

[13] Jia Yu, Jinxuan Wu, and Mohamed Sarwat. A demonstration of geospark: A cluster computing framework for processing big spatial data. In *ICDE*, pages 1410–1413, 2016.

[14] Johannes Niedermayer, Andreas Züfle, Tobias Emrich, Matthias Renz, Nikos Mamoulis, Lei Chen, and Hans-Peter Kriegel. Probabilistic Nearest Neighbor Queries on Uncertain Moving Object Trajectories. *PVLDB*, 7(3):205–216, 2013.

[15] Yi Xia, Yirong Yang, and Yun Chi. Mining Association Rules with Non-uniform Privacy Concerns. In *DMKD*, pages 27–34, 2004.

[16] Thanh T. L. Tran, Charles A. Sutton, Richard Cocci, Yanming Nie, Yanlei Diao, and Prashant J. Shenoy. Probabilistic Inference over RFID Streams in Mobile Environments. In *ICDE*, pages 1096–1107, 2009.

[17] Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, and Andreas Züfle. Probabilistic Frequent Itemset Mining in Uncertain Databases. In *SIGKDD*, pages 119–128, 2009.

[18] Lizhen Wang, Jun Han, Hongmei Chen, and Junli Lu. Top-k probabilistic prevalent co-location mining in spatially uncertain data sets. *Frontiers of Computer Science*, 10(3):488–503, 2016.

[19] Mark Pogson and Pete Smith. Effect of spatial data resolution on uncertainty. *Environmental Modelling and Software*, 63:87–96, 2015.

[20] Liming Zhan, Ying Zhang, Wenjie Zhang, and Xuemin Lin. Finding top k most influential spatial facilities over uncertain objects. *IEEE Trans. Knowl. Data Eng.*, 27(12):3289–3303, 2015.

[21] L. Srikar Muppirisetty, Tommy Svensson, and Henk Wymeersch. Spatial wireless channel prediction under location uncertainty. *IEEE Trans. Wireless Communications*, 15(2):1031–1044, 2016.

[22] Yan Huang, Shashi Shekhar, and Hui Xiong. Discovering Colocation Patterns from Spatial Data Sets: A General Approach. *IEEE Trans. Knowl. Data Eng.*, 16(12):1472–1485, 2004.

[23] Lizhen Wang, Lihua Zhou, Joan Lu, and Jim Yip. An Order-clique-based Approach for Mining Maximal Co-Locations. *Inf. Sci.*, 179(19):3370–3382, 2009.

[24] Jin Soung Yoo and Mark Bow. Mining Top-k Closed Co-location Patterns. In *ICSDM*, pages 100–105, 2011.

[25] Byoungyoung Lee, Jinoh Oh, Hwanjo Yu, and Jong Kim. Protecting Location Privacy Using Location Semantics. In *SIGKDD*, pages 1289–1297, 2011.

[26] Claudio Agostino Ardagna, Marco Cremonini, Ernesto Damiani, Sabrina De Capitani di Vimercati, and Pierangela Samarati. Location Privacy Protection Through Obfuscation-Based Techniques. In *DBSec*, pages 47–60, 2007.

[27] Xin Zhang, Nikos Mamoulis, David W. Cheung, and Yutao Shou. Fast Mining of Spatial Collocations. In *KDD*, pages 384–393, 2004.

[28] Hui Xiong, Shashi Shekhar, Yan Huang, Vipin Kumar, Xiaobin Ma, and Jin Soung Yoo. A Framework for Discovering Co-Location Patterns in Data Sets with Extended Spatial Objects. In *SDM*, pages 78–89, 2004.

[29] Jin Soung Yoo, Shashi Shekhar, and Mete Celik. A Join-Less Approach for Co-Location Pattern Mining: A Summary of Results. In *ICDM*, pages 813–816, 2005.

[30] Jin Soung Yoo and Shashi Shekhar. A Joinless Approach for Mining Spatial Colocation Patterns. *IEEE Trans. Knowl. Data Eng.*, 18(10):1323–1337, 2006.

[31] Mete Celik, James M Kang, and Shashi Shekhar. Zonal Co-location Pattern Discovery with Dynamic Parameters. In *ICDM*, pages 433–438, 2007.

[32] Sajib Barua and Jörg Sander. SSCP: Mining Statistically Significant Co-location Patterns. In *SSTD*, pages 2–20, 2011.

[33] Feng Qian, Qinming He, Kevin Chiew, and Jiangfeng He. Spatial Co-location Pattern Discovery without Thresholds. *Knowl. Inf. Syst.*, 33(2):419–445, 2012.

[34] Johannes Niedermayer, Andreas Züfle, Tobias Emrich, Matthias Renz, Nikos Mamoulis, Lei Chen, and Hans-Peter Kriegel. Probabilistic Nearest Neighbor Queries on Uncertain Moving Object Trajectories. *PVLDB*, 7(3):205–216, 2013.

[35] Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, and Andreas Züfle. Probabilistic Frequent Itemset Mining in Uncertain Databases. In *KDD*, pages 119–128, 2009.

[36] Lizhen Wang, Pingping Wu, and Hongmei Chen. Finding Probabilistic Prevalent Colocations in Spatially Uncertain Data Sets. *IEEE Trans. Knowl. Data Eng.*, 25(4):790–804, 2013.

[37] Zhi Liu and Yan Huang. Mining Co-locations under Uncertainty. In *SSTD*, pages 429–446, 2013.

[38] Mike Y. Chen, Timothy Sohn, Dmitri Chmelev, Dirk Hähnel, Jeffrey Hightower, Jeff Hughes, Anthony LaMarca, Fred Potter, Ian E. Smith, and Alex Varshavsky. Practical Metropolitan-Scale Positioning for GSM Phones. In *UbiComp*, pages 225–242, 2006.

[39] Dieter Pfoser and Christian S. Jensen. Capturing the Uncertainty of Moving-Object Representations. In *SSD*, pages 111–132, 1999.

[40] Tingting Dong, Chuan Xiao, Xi Guo, and Yoshiharu Ishikawa. Processing Probabilistic Range Queries over Gaussian-Based Uncertain Data. In *SSTD*, pages 410–428, 2013.

[41] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[42] Yasuhiko Morimoto. Mining Frequent Neighboring Class Sets in Spatial Databases. In *SIGKDD*, pages 353–358, 2001.

[43] Jin Soung Yoo and Shashi Shekhar. A Partial Join Approach for Mining Co-location Patterns. In *GIS*, pages 241–249, 2004.

[44] Lizhen Wang, Yuzhen Bao, Joan Lu, and Jim Yip. A New Join-less Approach for Co-location Pattern Mining. In *CIT*, pages 197–202, 2008.

[45] Yan Huang, Hui Xiong, Shashi Shekhar, and Jian Pei. Mining Confident Colocation Rules without A Support Threshold. In *SAC*, pages 497–501, 2003.

[46] Feng Qian, Kevin Chiew, Qinming He, Hao Huang, and Lianhang Ma. Discovery of Regional Co-location Patterns with k-Nearest Neighbor Graph. In *PAKDD (1)*, pages 174–186, 2013.

[47] Robert Munro, Sanjay Chawla, and Pei Sun. Complex Spatial Relationships. In *ICDM*, pages 227–234, 2003.

[48] Shisheng Yang, Lizhen Wang, Xuguang Bao, and Junli Lu. A framework for mining spatial high utility co-location patterns. In *12th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015, Zhangjiajie, China, August 15-17, 2015*, pages 595–601, 2015.

[49] Jundong Li, Aibek Adilmagambetov, Mohomed Shazan Mohomed Jabbar, Osmar R. Zaïane, Alvaro Osornio-Vargas, and Osnat Wine. On discovering co-location patterns in datasets: a case study of pollutants and child cancers. *GeoInformatica*, 20(4):651–692, 2016.

[50] Wenhao Yu. Spatial co-location pattern mining for location-based services in road networks. *Expert Syst. Appl.*, 46:324–335, 2016.

[51] Reynold Cheng, Tobias Emrich, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, Goce Trajcevski, and Andreas Züfle. Managing uncertainty in spatial and spatio-temporal data. In *ICDE*, pages 1302–1305, 2014.

[52] Chun Kit Chui, Ben Kao, and Edward Hung. Mining Frequent Itemsets from Uncertain Data. In *PAKDD*, pages 47–58, 2007.

[53] Charu C. Aggarwal, Yan Li, Jianyong Wang, and Jing Wang. Frequent pattern mining with uncertain data. In *KDD*, pages 29–38, 2009.

[54] Liwen Sun, Reynold Cheng, David W. Cheung, and Jiefeng Cheng. Mining Uncertain Data with Probabilistic Guarantees. In *KDD*, pages 273–282, 2010.

[55] Yongxin Tong, Lei Chen, Yurong Cheng, and Philip S. Yu. Mining Frequent Itemsets over Uncertain Databases. *PVLDB*, 5(11):1650–1661, 2012.

[56] R.B.J.T. Allenby and Alan Slomson. How to Count: An Introduction to Combinatorics. In *Discrete Mathematics and Its Applications (2ed.)*. CRC Press, 2010.

[57] Yoshiharu Ishikawa, Yuichi Iijima, and Jeffrey Xu Yu. Spatial Range Querying for Gaussian-Based Imprecise Query Objects. In *ICDE*, pages 676–687, 2009.

[58] Mihael Ankerst, Bernhard Braunmüller, Hans-Peter Kriegel, and Thomas Seidl. Improving Adaptable Similarity Query Processing by Using Approximations. In *VLDB*, pages 206–217, 1998.

[59] Feifei Li, Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-Hua Teng. On Trip Planning Queries in Spatial Databases. In *SSTD*, pages 273–290, 2005.

[60] Yan Huang, Jian Pei, and Hui Xiong. Mining Co-Location Patterns with Rare Events from Spatial Data Sets. *GeoInformatica*, 10(3):239–260, 2006.

[61] Lizhen Wang, Lihua Zhou, Joan Lu, and Jim Yip. An Order-clique-based Approach for Mining Maximal Co-locations. *Inf. Sci.*, 179(19):3370–3382, September 2009.

[62] Dong Xin, Jiawei Han, Xifeng Yan, and Hong Cheng. Mining Compressed Frequent-Pattern Sets. In *VLDB*, pages 709–720, 2005.

[63] Guimei Liu, Haojun Zhang, and Limsoon Wong. Finding Minimum Representative Pattern Sets. In *KDD*, pages 51–59, 2012.

[64] Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. Summarizing Itemset Patterns: A Profile-based Approach. In *KDD*, pages 314–323, 2005.

[65] Roberto J. Bayardo Jr. Efficiently Mining Long Patterns from Databases. In *SIGMOD*, pages 85–93, 1998.

[66] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In *ICDT*, pages 398–416, 1999.

[67] Toon Calders and Bart Goethals. Mining All Non-derivable Frequent Itemsets. In *PKDD*, pages 74–85, 2002.

[68] Natwar Modani and Kuntal Dey. Large Maximal Cliques Enumeration in Sparse Graphs. In *CIKM*, pages 1377–1378, 2008.

[69] Song Wang, Yan Huang, and Xiaoyang Sean Wang. Regional Co-locations of Arbitrary Shapes. In *SSTD*, pages 19–37, 2013.

[70] Claudio Silvestri, Francesco Cagnin, Francesco Lettich, Salvatore Orlando, and Monica Wachowicz. Mining condensed spatial co-location patterns. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS 2015, Bellevue, WA, USA, November 3-6, 2015*, pages 84–87, 2015.

[71] Xuguang Bao, Lizhen Wang, and Hongmei Chen. Ontology-based interactive post-mining of interesting co-location patterns. In *Web Technologies and Applications - 18th Asia-Pacific Web Conference, APWeb 2016, Suzhou, China, September 23-25, 2016. Proceedings, Part II*, pages 406–409, 2016.

[72] Ghazi Al-Naymat. Enumeration of Maximal Clique for Mining Spatial Co-location Patterns. In *AICCSA*, pages 126–133, 2008.

[73] James Cheng, Linhong Zhu, Yiping Ke, and Shumo Chu. Fast Algorithms for Maximal Clique Enumeration with Limited Memory. In *KDD*, pages 1240–1248, 2012.

[74] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. MIT Press, 2001.

[75] Leyla Kazemi and Cyrus Shahabi. Geocrowd: Enabling query answering with spatial crowdsourcing. In *SIGSPATIAL*, pages 189–198, 2012.

[76] Yongjian Zhao and Qi Han. Spatial crowdsourcing: Current state and future directions. *IEEE Communications Magazine*, 54(7):102–107, 2016.

[77] Yongxin Tong, Jieying She, Bolin Ding, Libin Wang, and Lei Chen. Online mobile micro-task allocation in spatial crowdsourcing. In *ICDE*, pages 49–60, 2016.

[78] Layla Pournajaf, Li Xiong, Vaidy S. Sunderam, and Slawomir Goryczka. Spatial Task Assignment for Crowd Sensing with Cloaked Locations. In *MDM*, pages 73–82, 2014.

[79] Hien To, Gabriel Ghinita, and Cyrus Shahabi. A Framework for Protecting Worker Location Privacy in Spatial Crowdsourcing. *PVLDB*, 7(10):919–930, 2014.

[80] Sangho Lee, Jong Kim, and Yoonho Kim. Preserving source- and sink-location privacy in sensor networks. *Comput. Sci. Inf. Syst.*, 13(1):115–130, 2016.

[81] Wei Wang, Yingjie Chen, and Qian Zhang. Privacy-preserving location authentication in wi-fi networks using fine-grained physical layer signatures. *IEEE Trans. Wireless Communications*, 15(2):1218–1225, 2016.

[82] Rong Yu, Jiawen Kang, Xumin Huang, Shengli Xie, Yan Zhang, and Stein Gjessing. Mixgroup: Accumulative pseudonym exchanging for location privacy enhancement in vehicular social networks. *IEEE Trans. Dependable Sec. Comput.*, 13(1):93–105, 2016.

[83] Hui Zhu, Rongxing Lu, Cheng Huang, Le Chen, and Hui Li. An efficient privacy-preserving location-based services query scheme in outsourced cloud. *IEEE Trans. Vehicular Technology*, 65(9):7729–7739, 2016.

[84] Feilong Tang, Jie Li, Ilsun You, and Minyi Guo. Long-term location privacy protection for location-based services in mobile cloud computing. *Soft Comput.*, 20(5):1735–1747, 2016.

[85] Shuang Zhao, Xiapu Luo, Bo Bai, Xiaobo Ma, Wei Zou, Xinliang Qiu, and Man Ho Au. I know where you all are! exploiting mobile social apps for large-scale location privacy probing. In *ACISP*, pages 3–19, 2016.

[86] Lichun Li, Rongxing Lu, and Cheng Huang. EPLQ: efficient privacy-preserving location-based query over outsourced encrypted data. *IEEE Internet of Things Journal*, 3(2):206–218, 2016.

[87] Pierangela Samarati. Protecting Respondents' Identities in Microdata Release. *IEEE Trans. Knowl. Data Eng.*, 13(6):1010–1027, 2001.

[88] Bugra Gedik and Ling Liu. Protecting Location Privacy with Personalized k-Anonymity: Architecture and Algorithms. *IEEE Trans. Mob. Comput.*, 7(1):1–18, 2008.

[89] Jie Hu, Liusheng Huang, Lu Li, Mingyu Qi, and Wei Yang. Protecting location privacy in spatial crowdsourcing. In *APWeb*, pages 113–124, 2015.

[90] Bhuvan Bamba, Ling Liu, Péter Pesti, and Ting Wang. Supporting Anonymous Location Queries in Mobile Environments with Privacy Grid. In *WWW*, pages 237–246, 2008.

[91] Marco Gruteser and Dirk Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *MobiSys*, pages 31–42, 2003.

[92] Bugra Gedik and Ling Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *ICDCS*, pages 620–629, 2005.

[93] Gabriel Ghinita, Panos Kalnis, and Spiros Skiadopoulos. PRIVE: Anonymous Location-based Queries in Distributed Mobile Systems. In *WWW*, pages 371–380, 2007.

[94] Agusti Solanas, Josep Domingo-Ferrer, and Antoni Martínez-Ballesté. Location Privacy in Location-Based Services: Beyond TTP-based Schemes. In *PiLBA*, 2008.

[95] Josep Domingo-Ferrer. Microaggregation for Database and Location Privacy. In *NGITS*, pages 106–116, 2006.

[96] Chi-Yin Chow, Mohamed F. Mokbel, and Xuan Liu. A Peer-to-peer Spatial Cloaking Algorithm for Anonymous Location-based Service. In *GIS*, pages 171–178, 2006.

[97] Gianluca Dini and Pericle Perazzo. Uniform Obfuscation for Location Privacy. In *DBSec*, pages 90–105, 2012.

[98] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. Private Queries In Location Based Services: Anonymizers Are Not Necessary. In *SIGMOD*, pages 121–132, 2008.

[99] Xun Yi, Russell Paulet, Elisa Bertino, and Vijay Varadharajan. Practical k nearest neighbor queries with location privacy. In *ICDE*, pages 640–651, 2014.

[100] Yao Shen, Liusheng Huang, Lu Li, Xiaorong Lu, Shaowei Wang, and Wei Yang. Towards preserving worker location privacy in spatial crowdsourcing. In *2015 IEEE Global Communications Conference, GLOBECOM 2015, San Diego, CA, USA, December 6-10, 2015*, pages 1–6, 2015.

[101] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. Differentially Private Grids for Geospatial Data. In *ICDE*, pages 757–768, 2013.

[102] Eirini C. Micheli, Giorgos Margaritis, and Stergios V. Anastasiadis. Efficient multi-user indexing for secure keyword search. In *EDBT/ICDT*, pages 390–395, 2014.

[103] Bo Cheng, Li Zhuo, Yu Bai, Yuanfan Peng, and Jing Zhang. Secure index construction for privacy-preserving large-scale image retrieval. In *BDCloud*, pages 116–120, 2014.

[104] Sun-Ho Lee and Im-Yeong Lee. A secure index management scheme for providing data sharing in cloud storage. *JIPS*, 9(2):287–300, 2013.

[105] Bijit Hore, Sharad Mehrotra, Mustafa Canim, and Murat Kantarcioglu. Secure Multidimensional Range Queries over Outsourced Data. *VLDB*, 21(3):333–358, 2012.

[106] Haibo Hu, Jianliang Xu, Chushi Ren, and Byron Choi. Processing Private Queries over Untrusted Data Cloud through Privacy Homomorphism. In *ICDE*, pages 601–612, 2011.

[107] Peng Wang and Chinya V. Ravishankar. Secure and Efficient Range Queries on Outsourced Databases using Rp-trees. In *ICDE*, pages 314–325, 2013.

[108] Wai Kit Wong, David Wai-Lok Cheung, Ben Kao, and Nikos Mamoulis. Secure kNN Computation on Encrypted Databases. In *SIGMOD*, pages 139–152, 2009.

[109] Xiang Cheng, Sen Su, Yiping Teng, and Ke Xiao. Enabling Secure and Efficient kNN Query Processing over Encrypted Spatial Data in the Cloud. *Security and Communication Networks*, 2, 2015.

[110] Hung Dang, Tuan Nguyen, and Hien To. Maximum Complex Task Assignment: Towards Tasks Correlation in Spatial Crowdsourcing. In *IIWAS*, page 77, 2013.

[111] Roula Karam and Michele Melchiori. A Crowdsourcing-Based Framework for Improving Geo-spatial Open Data. In *SMC*, pages 468–473, 2013.

[112] Peng Cheng, Xiang Lian, Zhao Chen, Rui Fu, Lei Chen, Jinsong Han, and Jizhong Zhao. Reliable Diversity-Based Spatial Crowdsourcing by Moving Workers. *PVLDB*, 8(10):1022–1033, 2015.

[113] Umair ul Hassan and Edward Curry. Efficient task assignment for spatial crowdsourcing: A combinatorial fractional optimization approach with semi-bandit learning. *Expert Syst. Appl.*, 58:36–56, 2016.

[114] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, and Linhong Zhu. Task selection in spatial crowdsourcing from worker's perspective. *GeoInformatica*, 20(3):529–568, 2016.

[115] Peng Cheng, Xiang Lian, Lei Chen, Jinsong Han, and Jizhong Zhao. Task assignment on multi-skill oriented spatial crowdsourcing. *IEEE Trans. Knowl. Data Eng.*, 28(8):2201–2215, 2016.

[116] Leyla Kazemi, Cyrus Shahabi, and Lei Chen. GeoTruCrowd: Trustworthy Query Answering with Spatial Crowdsourcing. In *SIGSPATIAL*, pages 304–313, 2013.

[117] Khanh-Hung Dang and Kim-Tuyen Cao. Towards Reward-based Spatial Crowdsourcing. In *ICCAIS*, pages 363–368, Nov 2013.

[118] Yousef Elmehdwi, Bharath K. Samanthula, and Wei Jiang. Secure k-nearest Neighbor Query over Encrypted Data in Outsourced Environments. In *ICDE*, pages 664–675, 2014.

[119] Oded Goldreich. *Foundations of Cryptography Volume 2 - Basic Applications*. University Press, Cambridge, 2004.

[120] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols - Techniques and Constructions*. Information Security and Cryptography. Springer, 2010.

[121] Bharath K. Samanthula, Fang-Yu Rao, Elisa Bertino, and Xun Yi. Privacy-Preserving Protocols for Shortest Path Discovery over Outsourced Encrypted Graph Data. In *IRI*, pages 427–434, 2015.

[122] An Liu, Kai Zheng, Lu Li, Guanfeng Liu, Lei Zhao, and Xiaofang Zhou. Efficient Secure Similarity Computation on Encrypted Trajectory Data. In *ICDE*, pages 66–77, 2015.

[123] Hans Delfs and Helmut Knebl. *Introduction to Cryptography: Principles and Applications*. Springer, 2002.

[124] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT*, pages 223–238, 1999.

[125] Jake Loftus and Nigel P. Smart. Secure Outsourced Computation. In *AFRICACRYPT*, pages 1–20, 2011.

[126] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption. In *STOC*, pages 1219–1234, 2012.

[127] Yi Sun, Qiaoyan Wen, Yudong Zhang, Hua Zhang, Zhengping Jin, and Wenmin Li. Two-Cloud-Servers-Assisted Secure Outsourcing Multiparty Computation. *The Scientific World Journal*, 2014:7, 2014.

[128] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM*, 18(9):509–517, 1975.

[129] Moni Naor and Benny Pinkas. Oblivious Transfer with Adaptive Queries. In *CRYPTO*, pages 573–590, 1999.

[130] Layla Pournajaf, Li Xiong, Vaidy S. Sunderam, and Slawomir Goryczka. Spatial Task Assignment for Crowd Sensing with Cloaked Locations. In *MDM*, pages 73–82, 2014.

[131] D. T. Lee and C. K. Wong. Worst-Case Analysis for Region and Partial Region Searches in Multidimensional Binary Search Trees and Balanced Quad Trees. *Acta Inf.*, 9:23–29, 1977.

Published Work

1. Bozhong Liu, Ling Chen, Xingquan Zhu, Ying Zhang, Chengqi Zhang and Weidong Qiu. Protecting Location Privacy in Spatial Crowdsourcing using Encrypted Data. In *EDBT*, 2017. Accepted.

2. Bozhong Liu, Ling Chen, Chunyang Liu, Chengqi Zhang and Weidong Qiu. Mining Co-locations from Continuously Distributed Uncertain Spatial Data. In *APWEB*, pages 66-78, 2016.

3. Bozhong Liu, Ling Chen, Chunyang Liu, Chengqi Zhang and Weidong Qiu. RCP Mining: Towards the Summarization of Spatial Co-location Patterns. In *SSTD*, pages 451-469, 2015.