Efficient Processing of Spatial Queries over Uncertain Database

by

Liming Zhan

B.E. Northeastern University P.R.C, 2000

M.E. Northeastern University P.R.C, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
Doctor of Philosophy
IN THE SCHOOL
OF
Computer Science and Engineering

UNSW
AUSTRALIA

Tuesday 10th February, 2015

PLEASE TYPE

THE UNIVERSITY OF NEW SOUTH WALES

Thesis/Dissertation Sheet

Surname or Family name: Zhan

First name: Liming Other name/s:

Abbreviation for degree as given in the University calendar: PhD

School: School of Computer Science and Engineering Faculty: Faculty of Engineering

Title: Efficient processing of spatial queries over uncertain database

Abstract 350 words maximum: (PLEASE TYPE)

Uncertainty is inherent in many important applications, and many important queries are re-investigated in the context of uncertain data models. Efficient algorithms are strongly demanded to analyze spatial uncertain data.

This thesis studies four fundamental problems to analyze spatial uncertain data by proposing efficient query processing algorithms, including (1) find top k influential facilities, (2) identify top k dominating objects, (3) range search on uncertain trajectories, and (4) top k similarity join.

Firstly, we study the problem of finding top k most influential facilities over uncertain objects. We propose a new ranking model to identify the top k most influential facilities, which captures influence of facilities on the uncertain objects. Effective and efficient algorithms are proposed following the filtering-verification paradigm by utilizing two uncertain object indexing techniques. To effectively support uncertain objects with a large number of instances, we further develop randomized algorithms with accuracy guarantee.

Secondly, we study the problem of top k dominating query on uncertain data, which is an essential method in the multi-criteria decision analysis when an explicit scoring function is not available. We formally introduce the top k dominating model, and propose effective and efficient algorithms to identify the top k dominating objects. Novel pruning techniques are proposed by utilizing the spatial indexing and statistic information to reduce CPU and I/O costs.

Thirdly, we investigate the problem of range search on uncertain trajectories by assuming uncertain trajectories are modeled by the Markov Chains. We propose a general framework for range search on uncertain trajectories following the filtering-refinement paradigm where summaries of uncertain trajectories are constructed to facilitate the filtering process. Statistics based and partition based filtering techniques are developed to enhance the filtering capabilities.

Finally, we investigate the problem of top k similarity join over multi-valued objects. We apply two types of quantile based distance measures to explore the relative instance distribution among the multiple instances of objects. Following a filtering-refinement framework, efficient and effective techniques to process top k similarity joins over multi-valued objects are developed. Novel distance, statistic and weight based pruning techniques are proposed to speed up the computations.

Declaration relating to disposition of project thesis/dissertation

I hereby grant to the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or in part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all property rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstracts International (this is applicable to doctoral theses only).

…… …… ……

Signature Witness Date

The University recognises that there may be exceptional circumstances requiring restrictions on copying or conditions on use. Requests for restriction for a period of up to 2 years must be made in writing. Requests for a longer period of restriction may be considered in exceptional circumstances and require the approval of the Dean of Graduate Research.

FOR OFFICE USE ONLY Date of completion of requirements for Award:

THIS SHEET IS TO BE GLUED TO THE INSIDE FRONT COVER OF THE THESIS

Copyright Statement

Signed .....

Date .....

Authenticity Statement

'I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format.?

Signed .....

Date .....

Originality Statement

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed .....

Date .....

Abstract

Uncertainty is inherent in many important applications, such as data integration, environmental surveillance, location-based services (LBS), global positioning system (GPS), sensor monitoring, radio-frequency identification (RFID) and moving objects management. In recent years, we have witnessed significant research efforts devoted to producing probabilistic database management systems, and many important queries are re-investigated in the context of uncertain data models. Efficient algorithms are strongly demanded in these applications to analyze spatial uncertain data.

This thesis studies four fundamental problems to analyze spatial uncertain data by proposing efficient query processing algorithms. These problems include (1) find top k influential facilities, (2) identify top k dominating objects, (3) range search on uncertain trajectories, and (4) top k similarity join.

Firstly, we study the problem of finding top k most influential facilities over a set of uncertain objects, which is an important and fundamental spatial query in the above applications. Based on the maximal utility principle, we propose a new ranking model to identify the top k most influential facilities, which carefully captures influence of facilities on the uncertain objects. By utilizing two uncertain object indexing techniques, R-tree and U-Quadtree, effective and efficient algorithms are proposed following the filtering and verification paradigm, which significantly

improves the performance of the algorithms in terms of CPU and I/O costs. To effectively support uncertain objects with a large number of instances, we further develop randomized algorithms with accuracy guarantee.

Secondly, we study the problem of top k dominating query on multi-dimensional uncertain objects, which is an essential method in the multi-criteria decision analysis when an explicit scoring function is not available. Particularly, we formally introduce the top k dominating model based on the state-of-the-art top k semantic over uncertain data. We also propose effective and efficient algorithms to identify the top k dominating objects. Novel pruning techniques are proposed by utilizing the spatial indexing and statistic information, which significantly improve the performance of the algorithms by reducing CPU and I/O costs.

Thirdly, we investigate the problem of range search on uncertain trajectories by assuming uncertain trajectories are modeled by the popular Markov Chains. In particular, we propose a general framework for range search on uncertain trajectories following the filtering and refinement paradigm where summaries of uncertain trajectories are constructed to facilitate the filtering process. Moreover, statistics based and partition based filtering techniques are developed to enhance the filtering capabilities.

Finally, we investigate the problem of top k similarity join over multi-valued objects. We apply two types of quantile based distance measures, ϕ-quantile distance and ϕ-quantile group-base distance, to explore the relative instance distribution among the multiple instances of objects. Then, following a filtering and refinement framework, efficient and effective techniques to process top k similarity joins over multi-valued objects are developed. Novel distance, statistic and weight based pruning techniques are proposed to speed up the computation.

Publications Involved in this Thesis

- **Liming Zhan**, Ying Zhang, Wenjie Zhang, Xuemin Lin. Finding Top k Most Influential Spatial Facilities over Uncertain Objects. CIKM 2012, *ERA Ranking A* (Chapter 3)

- **Liming Zhan**, Ying Zhang, Wenjie Zhang, Xuemin Lin. Finding Top k Most Influential Spatial Facilities over Uncertain Objects. Under review by TKDE (Chapter 3)

- **Liming Zhan**, Ying Zhang, Wenjie Zhang, Xuemin Lin. Identifying Top k Dominating Objects over Uncertain Data. DASFAA 2014, *ERA Ranking A* (Chapter 4)

- **Liming Zhan**, Ying Zhang, Wenjie Zhang, Xiaoyang Wang, Xuemin Lin. Range Search on Uncertain Trajectories. To be submitted for a major conference or journal (Chapter 5)

- Wenjie Zhang, **Liming Zhan**, Ying Zhang, Muhammad Aamir Cheema, Xuemin Lin. Efficient Top-k Similarity Join Processing over Multi-valued Objects. World Wide Web 2014, *ERA Ranking A* (Chapter 6)

Dedication

To my family

my parents

my relatives

my friends

all support me

For their love and support

Acknowledgements

First and foremost, I would like to present my great gratitude to my supervisor Prof. Xuemin Lin for his support and guidance. I really appreciate his insightful advices and suggestions during my Ph.D. study. Also, I learnt the characteristics of an excellent researcher from Prof. Xuemin Lin - passion, diligence and earnestness in the research.

Secondly, I would like to express my heartfelt gratitude to my joint-supervisor, Dr. Ying Zhang for his constant encouragement and guidance. He has walked me through all the stages of the writing of this thesis. Without his consistent and illuminating instruction, this thesis could not have reached its present form.

Thirdly, parts of the work in this thesis were conducted in collaboration with Dr. Wenjie Zhang, Dr. Muhammad Aamir Cheema, Mr. Xiaoyang Wang. I thank them for supporting the work presented in this thesis, and much more.

Besides, special thanks go to the former group members: Dr. Gaoping Zhu, Dr. Ke Zhu, Dr. Zhitao Shen, Dr. Xiang Zhao, Dr. Wenren Yu, Dr. Chuan Xiao, and Dr. Haichuan Shang as well as fellow group members: Dr. Lijun Chang, Dr. Lu Qin, Mr. Shiyu Yang, Mr. Chengyuan Zhang, Mr. Xing Feng, Ms. Shenlu Wang, Mr. Xiang Wang, Mr. Longbin Lai, Mr. Long Yuan, Mr. Jianye Yang, and Mr. Yang Wang. The time we spent together will be memorized forever.

Last but not least, I am very thankful to my beloved family, parents and relatives for their love, support and encouragement, and for being with me always in my life.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Database systems are widely used today. Almost every information system contains one or more databases, and almost every commercial product is based on database system. From a traditional perspective, databases are used to store precise values. However, uncertain data are inherent in many applications such as environmental surveillance, market analysis, sensor network monitoring, radio-frequency identification (RFID), location-based services (LBS), and moving object management. The uncertain data in those applications are generally caused by data randomness and incompleteness, limitation of measuring equipment, delayed data updates, etc. Recently, considerable research efforts have been devoted into the field of uncertainty-aware spatial query processing such that the uncertainty of the data can be effectively and efficiently tackled. Thus, many query types have been reinvestigated under the uncertain semantics, including query evaluation [DS07], aggregate queries [BDJ+07], spatial joins [CSP+06], top k queries [SIC07], skyline queries [PJLY07, LZZC11], dominating queries [LC09b, ZLZ+10a], nearest neighbor queries [CKP04], reverse nearest neighbor queries [CLW+10], range queries [TXC07, ZZLL12], etc.

This thesis focus on developing efficient solution to spatial queries on uncertain database. In particular, the following four problems are comprehensively investigated:

1) Find Top k Influential Facilities.

2) Identify Top k Dominating Objects.

3) Range Search on Uncertain Trajectories.

4) Top k Similarity Join

In this chapter, we first motivate the problems, and unveil the challenges to each of them, respectively, in four subsections. Then, we explain the major contributions of our solutions and how the whole thesis is organized.

1.1 Motivations

1.1.1 Find Top k Influential Facilities

Bichromatic reverse nearest neighbor (BRNN) query has been extensively studied as an important spatial operator ever since it was introduced in [KM00] due to a wide spectrum of applications such as decision support, profile-based marketing, resource allocation, etc. Informally, given a set $\mathcal{F}$ of facilities (e.g., gas station, supermarket and warehouse) and a set of $\mathcal{O}$ of objects (e.g., car, person), the influence of a particular facility F can be evaluated by the number of bichromatic reverse nearest neighbors of F, e.g., the number of objects whose nearest neighbors are F regarding $\mathcal{F}$. Intuitively, it is desirable to identify the most influential facilities for various reasons such as resource allocation and decision making. Motivated by this, some existing work (e.g., [XZKD05]) proposed efficient algorithms

to identify the influential facilities, in which they assume the location of an object or facility is precisely described by a spatial point. As shown in Figure 1.1(a), we can calculate the influence score (e.g., the number of reverse nearest neighbors) for each facility, where F_1, F_2 and F_3 have scores 1, 2 and 0 respectively. Nevertheless, in many applications the location of an object may be uncertain due to various reasons such as data randomness and incompleteness, the limitation of measuring equipment, delay or loss of data updates and privacy preservation. Following are two motivating examples.

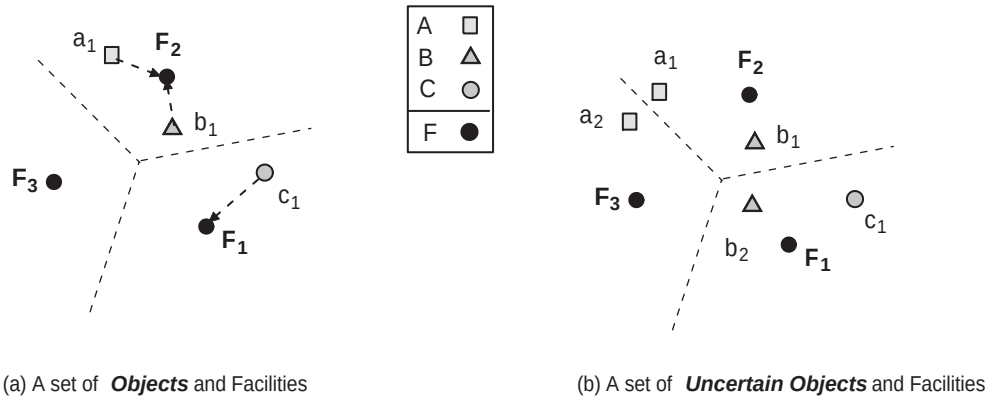


Figure 1.1: Influential Facilities Example

Motivating Examples. In some warehouse management systems, the RFID tags are attached to the items and their current locations can be obtained by RFID readers. Since the RFID reading may be noisy due to the sensitivity of the low cost readers to various environmental factors such as interference from nearby metal objects and contention among tags, the location of an object may be modeled as an uncertain object which is described by multiple instances. For instance, in Figure 1.1(b), the item A may appear at two positions a_1 and a_2 with the same probability. The facilities in Figure 1.1(b) represent the dispatching points for various items. Suppose an item will be delivered to the closest dispatch point. For a proper resource (e.g., labor, truck) allocation, the manager may want to know

the k dispatching points with the highest influences (workload).

Another example application is the location based service (LBS). The location of a mobile user can be described as an uncertain object, since her/his location may be derived based on the nearest contour lines of user's possible locations regarding the nearby towers. In Figure 1.1(b), the mobile users are uncertain objects and supermarkets correspond to the facilities. Suppose that users tend to visit the nearby supermarket, it is meaningful to find the top k most promising supermarkets, i.e., supermarkets which influence the largest number of users.

Challenges. Motivated by the above examples, it is desirable to study the problem of finding top k most influential facilities over uncertain objects. The challenges are three-fold.

Firstly, unlike the traditional spatial database in which an object only contributes to the influence score of one facility[1], it is non-trivial to evaluate the influence of a facility because of the existence of multiple instances. As shown in Figure 1.1(b), the uncertain object A may be influenced by both F_2 and F_3. Therefore, when we rank the influences of the facilities, it is desirable to propose new model to capture the uncertainty of the uncertain objects.

The second challenge is the efficiency of the algorithm. Computing the top k most influential facilities over uncertain objects is much more complicated than that of traditional objects (points). Although our new model can avoid enumerating all possible worlds, the computation cost is still expensive if we conduct the calculation in a straightforward way due to the existence of multiple instances of uncertain objects.

The third challenge arises from the possible massive number of instances in each uncertain object. In many applications, the uncertainty of an object may

[1]Suppose the ties are broken arbitrarily if there are multiple nearest facilities for an object.

be described by a continuous probability density function, and it is a common practice to discretize the continuous distribution by sampling method. As shown in [TXC07], a large number of instances are necessary to properly capture the probability distribution of the uncertain objects. On the other hand, the Monte-Carlo simulation has been widely applied to tackle the complicated correlation among uncertain objects in many recent works (e.g., [JXW+08]), and hence each uncertain object will be represented by a set of sampled instances. Consequently, it is desirable to develop efficient randomized algorithms based on samples of the uncertain objects.

1.1.2 Identify Top k Dominating Objects

Ranking query is an essential analytic method which focuses on retrieving the top k most important answers from massive quantity of data according to an user's preference. In many applications, users need to make decision against multiple features of the objects. For instance, cost, comfort, safety, and fuel economy may be some of the main criteria we consider when purchasing a car. Therefore, each object can be described by a point in a multi-dimensional space where each dimension corresponds to a particular selection feature. If there is a scoring (utility) function (e.g., additive linear function) which can quantify the preference of a user, objects can be immediately ranked based on their corresponding scores. However, in many scenarios users cannot find a proper scoring function due to various reasons such as the lack of domain knowledge. By utilizing the dominance relationship, the **top k dominating query** provides a simple and intuitive way to rank objects when an explicit scoring function is not available. The dominance relationship has been widely used in the multi-criteria decision analysis where an object A *dominates* another object B if A is not worse than B on every dimension and A

is strictly better than B on at least one dimension. The goodness of an object A, namely *dominance score*, can be naturally measured by the number of other objects dominated by A in a set of objects, and the top k objects with highest *dominance scores* are returned as the top k dominating objects.

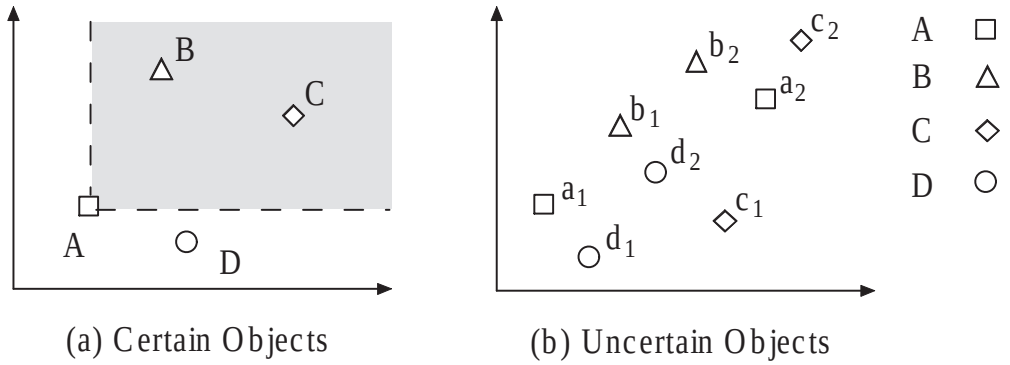


Figure 1.2: Certain Objects and Uncertain Objects

Example 1.1. *As shown in Figure 1.2(a), there is a set $\mathcal{O}$ of 4 objects (points). The dominance score of the object A is 2 which is the number of other points within the shaded region. Similarly, the dominance scores of B, C and D are 0, 0 and 1 respectively. Therefore, the results of the top 2 dominating query on $\mathcal{O}$ are A and D.*

Motivation. In many applications such as data integration, environmental surveillance, location based service, the uncertainty is inherent due to many factors including limitation of measuring equipments, probabilistic model based data integration, noise, delay or loss of data updates, etc. Consequently, the data in the above applications are usually described by probabilistic model (i.e., uncertain objects) instead of deterministic points in a multi-dimensional space where an uncertain object may be described by probabilistic density function or a set of instances (points). For example, in the meteorology system, sensors collect the temperature and relative humidity at a large number of sites. The readings may be uncertain due to the

noise or the limit of the sensor reader. Another example is the performance evaluation of NBA players. A player attends multiple games and statistics (e.g. score and #rebounds) of each game are recorded. Consequently, the performance of a player may be naturally modeled as an uncertain object where each record corresponds to an instance. In these applications, the top k dominating query plays an important role in multi-criteria decision analysis when the scoring function is not available. For instance, users may identify the top k risky observation sites in the meteorology system, or find the top k valuable players based on their game-by-game performance. Due to the inherent differences between uncertain data and traditional data, many important queries are re-investigated in the context of uncertain data models. These motivate us to study the problem of the top k dominating query on uncertain data.

Challenges. The main challenges of the problem are two-fold. Firstly, we need to develop a model to properly identify the top k dominating objects. Secondly, efficient computation algorithm is required to support the new top k dominating query on uncertain data.

As shown in Figure 1.2(b), each uncertain object may consist of multiple instances (points) , and each instance will appear with a particular probability. For example, the uncertain object A consists of two instance a_1 and a_2. Due to the existence of multiple instances of the uncertain objects, it is non-trivial to measure the number of other uncertain objects dominated by an uncertain object (i.e., *dominance score*) since the traditional dominance relationship is defined against two points. Intuitively, we can derive the *dominance score* of an instances u based on the probability mass of the instances which are dominated by u. Then the problem of the top k dominating query can be mapped to the problem of the top k query on uncertain data since each multi-dimensional uncertain object cor-

responds to a score distribution based on the *dominance score* and appearance probabilities of their instances. The problem of the top k query on uncertain data has been extensively studied in recent years, and many models are proposed such as *expected score*, U-top$k$[SIC07], U-$k$ranks [SIC07], PT-k [HPZL08b], Global-top k [ZC08], *expected rank* [CLY09], c-Typical-Top$k$ [GZM09], and *parameterized ranking* [LSD11]. Particularly, as shown in [LSD11] the *parameterized ranking* function can unify other popular ranking semantics, and hence it is widely used as the de facto top k semantics for uncertain data. Although there are some existing work [LC13, LC09b, ZLZ$^+$10a] investigating the top k dominating query on uncertain data, none of them supports the *parameterized ranking* semantics. Moreover, as shown in [CLY09] the top k semantics adopted in [LC13, LC09b, ZLZ$^+$10a] cannot properly capture the ranking of both probabilities and values. Therefore, it is desirable to formally define the top k dominating query on multi-dimensional uncertain objects, and develop efficient and effective algorithms to identify top k dominating uncertain objects.

1.1.3 Range Search on Uncertain Trajectories

With the rapid development of positioning technologies such as radio frequency identification (RFID), wireless sensor networks, smart-phone, radar, satellite and global positioning system (GPS), massive spatio-temporal data has been mounting up. Due to physical and resource limitations of the data collection devices, it is infeasible to continuously capture the location of a moving object (e.g., vehicle, people, animal and iceberg) for each point of time, and hence the uncertainty arises between two subsequent discrete observations. For instance, to save the energy and the communication cost, a taxi may report its location at a low frequency [ZZXZ12]. The time period between two check-in positions might be long in Geo-social appli-

cations such as bike routes[2] and tourist routes[3] [NZE+13]. Consequently, a large volume of spatio-temporal data with low sampling rate is described by *uncertain trajectories* where the possible locations of a moving object between two subsequent observations are captured by a time-dependent random variable (i.e., a stochastic process).

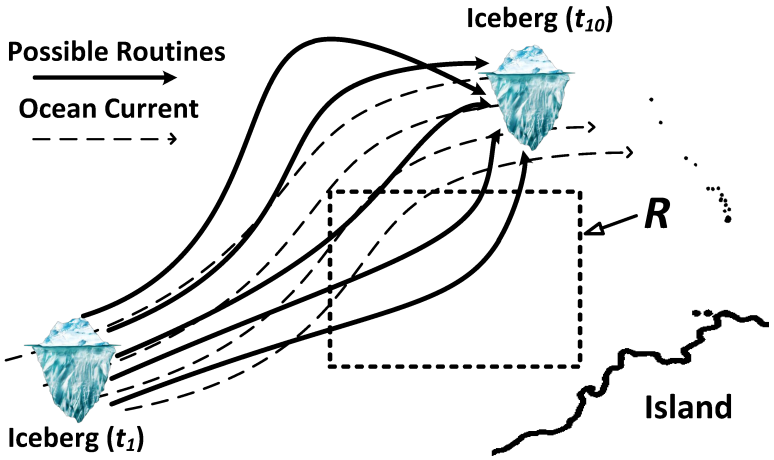


Figure 1.3: Uncertain Trajectory Example

Therefore, we investigate the problem of *range search* on uncertain trajectories, which is critical to make sense of uncertain trajectories in many key applications such as environment monitoring, location based service, traffic management, and national security. Informally, we aim to retrieve a set of moving objects (trajectories) which consistently appear within a given area with high probabilities during a particular time period. Below are two motivation examples for range search on uncertain trajectories.

Motivation Examples. In Figure 1.3, an iceberg is observed at times t_1 and t_{10} by satellite or radar systems, while its precise location is unknown at a time $t \in (t_1, t_{10})$ due to the resource limitation. Fortunately, according to the knowledge of nearby ocean currents as well as the historical iceberg trajectory data, an expert can derive

[2] `http://www.bikely.com/`

[3] `http://www.everytrail.com/`

possible locations of this iceberg based on Markov Chain model [EKM+12b]. To choose a particular region for setting up an oil platform or deploying a major military excises, we may need to evaluate to what degree a region (e.g, R in Figure 1.3) is affected by icebergs during a certain period in history.Since the location of an iceberg might be uncertain, we can ignore an iceberg at time t if the likelihood of this iceberg falling in the region R is smaller than a given threshold θ ($0 < \theta \leq 1$). Moreover, we may only be interested in the icebergs which consistently (say, at least η times) appear within the region with probability at least θ, which is the range search on uncertain trajectories. Similarly, in the study of wild animal migration, the range search on uncertain trajectories can help scientists to evaluate the importance of a region during a period of time, where locations of a tagged animal can be observed by wireless RFID sensors from time to time.

Challenges. Same as [EKM+12b, EKM+12a, NZE+13, XGC+13], we assume the uncertain trajectories are described by Markov Chain model because it has solid theoretical foundation and rich applications. A straightforward approach for the problem of range search on uncertain trajectories is to calculate the *appearance probability* of each moving object o at each time within the query time interval, and count the number of times in which o appears within the search region with probability at least θ. Then an object is qualified if the number of accumulated times exceeds a duration threshold η. However, as reported in [EKM+12a], the computation is still very expensive although efficient algorithm is developed in [EKM+12b]. Then, it is desirable to develop efficient and effective indexing and pruning algorithms to reduce the computations.

1.1.4 Top k Similarity Join

The need of similarity join over multi-valued objects stems from many important applications. In geographic information system (GIS), a group of simple spatial objects may be evaluated as a whole [KN96, RSV01]. For instance, to evaluate a community, a real estate development company may model it as a multi-valued object and each instance corresponds to a property with some feature values such as property price, household income, distance to beach, distances to living facilities, etc. A top k similarity join may be issued to identify the most similar communities from two large cities or from two countries, such that the price fluctuation of one community could be used as a mirror to the management of another one. Similarly, in sports, the performance of a player may be described by her game-to-game statistics in various games. So each player could be represented by a multi-valued object where each instance corresponds to her statistics, such as heights and number of trials in high-jump, in a particular game she attended. A similarity join over two sets of players may help to retrieve players with similar performances. Hence, the successful career path of one player provides a prediction of the success of her counterpart in coming competitions.

Similarity join is also a fundamental and crucial analyzing tool in internet and web information systems. In online shopping systems, it is interesting to retrieve similar pairs of shops or sellers where a shop or seller is modeled by a set of retail items with various features including item type and price range. Here a shop or seller could be modeled as a multi-valued object where each instance corresponds to a retail item. Identifying similar communities from online social networks is another important application of similarity join over multi-valued objects where one community (modeled as a multi-valued object) consists a set of individuals (instances) [WQWZ09, MBJ13].

Motivation Examples. The existing model for handling similarity joins over objects with multiple instances follows the probabilistic semantics on uncertain objects [CSP+06, KKPR06, LS08] and aims to capture relative instance distribution among objects with multiple instances. Nevertheless, uncertain objects are inherently different than multi-valued objects. Instances of an uncertain object are mutually exclusive which means at most one instance can appear at a particular time, while all the values/instances of a multi-valued object must occur simultaneously at any time. Moreover, as shown in [ZLC+10], models based on uncertain semantics cannot always capture the relative distributions of multi-valued objects. Take the example in Figure 1.4. For simplicity we assume multi-valued object U_1 has only one instance with the value (score) of 10, while multi-valued objects V_1 and V_2 both have m instances spread between 9.0 to 9.99 as depicted in Figure 1.4(a). Each instance from the same object takes the same weight. Suppose we want to retrieve the top-1 similarity join result from $\{U_1\}$ and $\{V_1, V_2\}$, namely, retrieve the more similar one from V_1 and V_2 to U_1. Following the possible world semantics, it is easy to verify that both V_1 and V_2 have the same probability, $\frac{1}{2}$, to be the most similar one to U_1 if Euclidean distance is used as the similarity metrics. We permute the distribution in Figure 1.4(a) to the distribution in Figure 1.4(b), V_1 and V_2 still have the same probability. This example demonstrates that the probabilistic approaches following the possible world semantics are not able to capture the relative distributions of instances. Another direct solution is to utilize simple aggregates such as average. Nevertheless, such a simple aggregate will have the same problem as pointed above regarding Figure 1.4.

The example in Figure 1.4 demonstrates that the existing probabilistic model and simple aggregates may be insensitive to relative distributions of object instances. Quantiles [YMT06] provide a succinct summary of data distributions and

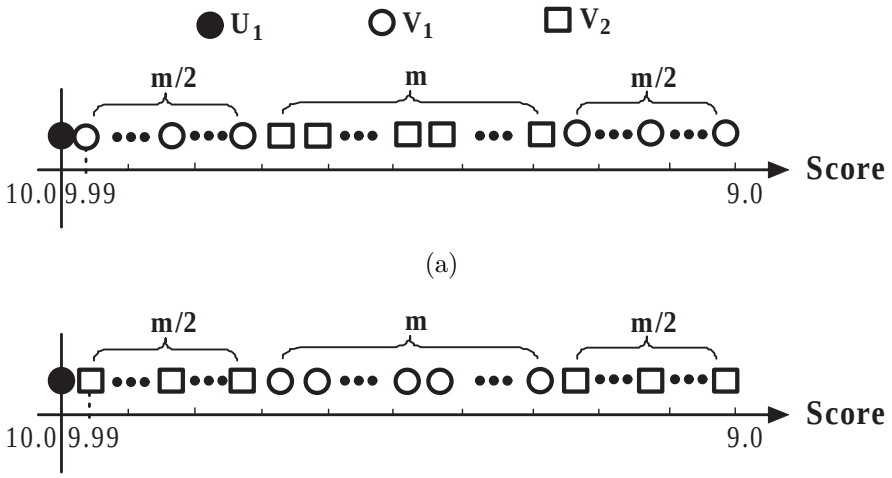


Figure 1.4: Motivating Example

is less sensitive to outliers.

Challenges. While the similarity between two conventional d-dimensional objects only involves two single points, identifying the most similar object pairs among multi-valued object sets involves multiple instances per object. Therefore, it is highly desirable to consider the relative instance distributions among multi-valued objects so that the similar pairs can be effectively retrieved. ϕ-quantile distance and ϕ-quantile group-base distance are first used for capturing instance distributions of multi-valued objects in [ZLC+10]. [ZLC+10] studies k-nearest neighbors (KNN) queries over multi-valued objects. Given a multi-valued query object Q and a set of multi-valued objects $\mathcal{U}$, a KNN query retrieves k objects from $\mathcal{U}$ with smallest quantile-based distance to Q. An immediate way to solve our problem can be conducted as follows. For each object $U \in \mathcal{U}$ (or $V \in \mathcal{V}$), we compute its KNN in $\mathcal{V}$ (or $\mathcal{U}$) using the techniques in [ZLC+10], and then select k most similar pairs based on the union of KNN results. Nevertheless, this involves the computation of KNN for each object in $\mathcal{U}$ (or each object in $\mathcal{V}$). Clearly, not every object in $\mathcal{U}$ (or $\mathcal{V}$) will be involved in the top k pairs since k is usually much smaller than

$min\{|\mathcal{U}|, |\mathcal{V}|\}$. Consequently, it is desirable to develop a set of novel, efficient, effective pruning techniques to prevent such redundant computation.

1.2 Contributions

In summary, we make the following contributions for the four problems studied in this thesis.

Find Top k Influence Facilities.

- Based on the *maximal utility principle*, we propose a new model to evaluate the influences of the facilities over a set of uncertain objects.
- Efficient R-tree and U-Quadtree based algorithms are presented following the *filtering and verification* paradigm. Novel pruning techniques are proposed to significantly improve the performance of the algorithms by reducing the number of uncertain objects and facilities in the computation.
- Efficient randomized techniques are proposed to provide approximate solution with accuracy guarantee.
- Comprehensive experiments demonstrate the effectiveness and efficiency of our techniques.

Identify Top k Dominating Objects.

- We present a general framework for the range search on uncertain trajectories following the filtering-and-refinement paradigm, where summaries of the objects are constructed to facilitate the filtering process.
- We propose effective and efficient top k dominating computation algorithms on multi-dimensional uncertain objects based on novel pruning techniques.

- We further improve the performance of the algorithm by utilizing statistics information of the uncertain objects.

- Comprehensive experiments on real and synthetic datasets demonstrate the efficiency and scalability of our techniques.

Range Search on Uncertain Trajectories.

- We formally define the problem of range search on uncertain trajectories.

- We present a general framework for the range search on uncertain trajectories following the filtering-and-refinement paradigm, where summaries of the objects are constructed to facilitate the filtering process.

- A simple and effective filtering technique is proposed based on statistics information of the uncertain trajectories.

- A partition based filtering technique is developed to further enhance the filtering capabilities. Effective summary construction algorithm is proposed based on some important observations.

- Comprehensive experiments on real-life and synthetic datasets demonstrate the effectiveness and efficiency of our techniques.

Top k Similarity Join

- We formalize the problem of top k similarity join over multi-valued objects, regarding two types of quantile-based distance metrics.

- Efficient and effective algorithms are developed to compute the top k similarity join results over two sets of multi-valued objects based on quantile-based distance metrics. Particularly, we propose novel and efficient distance,

statistic and weight based pruning techniques to significantly speed up the computation.

- Comprehensive experiments are conducted on both real and synthetic data to demonstrate the efficiency and effectiveness of our techniques. It also demonstrates that the techniques developed in this thesis are up to 2 orders of magnitude more efficient than naively applying KNN techniques in [ZLC+10].

1.3 Thesis Organization

We organize the rest of the thesis as follows. We study the four aforementioned problems in four chapters. Before delving into the details of our solutions, a survey of previous work on uncertain spatial queries and its variants are presented in Chapter 2. Then for each of the problem, we present the problem definition, preliminaries, proposed solution, empirical studies and summary in individual chapters. Chapter 3 investigates the problem of finding top k most influential spatial facilities over uncertain objects, Chapter 4 studies the problem of identifying top k domination objects over uncertain data, Chapter 5 discusses the problem of range search on uncertain trajectories, and Chapter 6 presents the approach to solve top k similarity join problem. Finally, Chapter 7 concludes the thesis and present feasible future work. Specifically, each chapter is structured as follows.

Chapter 2 provides a literature review of the existing work related to the four problems studied in this thesis. We introduce current uncertain models and query techniques in four parts.

- Uncertainty Models

- Probabilistic Queries

- Indexing Uncertain Objects
- Uncertain Trajectories

In Chapter 3, we study the problem of finding top k most influential spatial facilities over uncertain objects. We first formally define the problem based on the possible world semantics. Then, we propose a new ranking model to identify the top k most influential facilities based on the maximal utility principle. Thirdly, effective and efficient exact algorithms are proposed following the filter and verification paradigm by utilizing two uncertain uncertain indexing techniques, R-tree and U-Quadtree. Afterwards, randomized algorithms are further developed to efficiently support uncertain objects with a large number of instances, and the accuracy is guaranteed. Finally, the performance evaluation is reported and analyzed.

In Chapter 4, we study the problem of top k domination query on multi-dimension uncertain objects. We first formally introduce the top k dominating model and define the problem based on parameterized ranking function. Secondly, we propose effective and efficient algorithm following filtering and verification paradigm by utilizing the spatial indexing and statistic information. Finally, comprehensive experiments on real and synthetic datasets demonstrate the effectiveness and efficiency of the techniques.

In Chapter 5, we study the problem of range search on uncertain trajectories. We first formally introduce the uncertain trajectories modeled by Markov Chains and give the problem definition. Then, a general framework for range search on uncertain trajectories following the filtering-and-refinement paradigm is proposed. Thirdly, we propose the technique to facilitate the filtering process by utilizing summaries of uncertain trajectories. Moreover, we develop statistics based and partition based filtering techniques to enhance the filtering capabilities. The effectiveness and efficiency of the proposed techniques are empirically evaluated in the

end.

In Chapter 6, we study the problem of top k similarity join problem over multi-value objects based on a ϕ-quantile distance. We first formalize the problem of top k similarity join over multi-valued objects, regarding two types of quantile-based distance metrics, respectively. Then, efficient and effective algorithms with distance, statistic and weight based pruning techniques are developed to compute the top k similarity join results over two sets of multi-valued objects based on quantile-based distance metrics. Comprehensive experiments are conducted on both real and synthetic data to demonstrate the efficiency and effectiveness of our techniques.

Chapter 7 finally summarizes this thesis and provides a number of feasible directions for future studies.

Chapter 2

Related Work

This chapter provides a non-exhaust review of the existing literature related to spatial queries on uncertain database. More specifically, we first give a brief introduction of models for uncertain data in Section 2.1. Next, we summarize existing techniques on various probabilistic query types in Section 2.2 followed by an overview of the exist techniques on indexing uncertain objects in Section 2.3. Finally, we present the related work on uncertain trajectories in Section 2.4.

2.1 Uncertainty Models

Uncertainty is (almost) everywhere, which is often caused by the limited perception and understanding of reality as well as can be inherent in nature. As an example, consider a meteorology system that monitors the temperatures, humidity, UV indexes in a large number of regions. The corresponding readings are taken by sensors in local areas, and transmitted to a central database periodically (*e.g.* every 10 minutes). The database content may not exactly reflect the current atmospheric status and the actual temperature in a region may have changed after the last time to measure them. Also, some sensors may not get precise data due to the sake of

energy conservation. The uncertainty in the above examples is caused by delayed data updates, sources of imprecision include data randomness, limitation of measuring equipments, and so on. Therefore, information retrieval directly based on uncertain data is meaningless, since the result does not have any quality guarantees. Consider another example that the query "find the clients currently in a special area". Returning the objects whose previous reported positions satisfy the query is inadequate, because many objects may have entered or left the search region since they contacted the server last time. The uncertainty of an object can be specified by three models: fuzzy model [GUP06], evidence-oriented model [Lee92, LSS96] and probabilistic model [SBHW06]. In this thesis we focus on probabilistic models since it is is not only the most widely used but also it is the only model adopted in existing DBMSs for uncertainty analysis.

Recently, the possible world semantics [IJ84, AKG87, AKG91, SBHW06] are widely used to model uncertainty. A possible world is a possible snapshot that may be observed and carries an existence probability. A possible world contains a set of instances of uncertain objects, and at most one instance per object in a possible world. A comprehensive study of possible world semantics is investigated in [SIC07, HPZL08a, YLKS08], and the formal definition of possible world semantics is given below.

Given a set of uncertain objects $U = \{U_1, \ldots, U_n\}$, each uncertain object consists of a set of instances. By assuming all objects are independent, a possible world $W = \{u_1, \ldots, u_n\}$ is a set of instances with one instance from each uncertain object. The probability of W to appear is $P(W) = \prod_{i=1}^{n} p_{u_i}$. Let $\mathcal{W} = \{W_1, \ldots, W_m\}$ be the set of all possible worlds, then $\sum_{W \in \mathcal{W}} P(W) = 1$. An uncertain object may be described by either a continuous or a discrete case. In continuous cases, an uncertain object U may be described by a probability density function (PDF)

f_U such that $\int_{u \in U} f_U(u)du = 1$; Nevertheless, in many applications PDFs are not always available. Instead, an uncertain object U is represented by a set of instances such that each instance $u \in U$ has a probability $P(u)$ to appear. Such a representation is also referred as a discrete case, has the property that $0 < P(u) \leq 1$ and $\sum_{u \in U} P(u) = 1$. In this thesis, we focus on discrete case, because we can discretize a continuous probability density function (PDF) of an uncertain object by sampling methods.

2.2 Probabilistic Queries

We review existing work on important probabilistic queries in this subsection.

2.2.1 Top k Queries

With the emergence of many recent important and novel applications involving uncertain data, there has been a great deal of research attention dedicated to this field. Particularly, top k queries are important in analyzing uncertain data. Unlike a top k query over certain data which returns the k best alternatives according to a ranking function, a top k query against uncertain data has inherently more sophisticated semantics.

Soliman *et al.* [SIC07] first relate top k queries with uncertain data. They define two types of important queries - U-Top$k$ and U-kRank, regarding discrete cases.

- U-Top$k$ returns a set of k records which as a whole have the highest probability to be the top k results in all possible worlds. Let D be an uncertain data set with possible worlds space $\mathcal{W} = \{W_1, \ldots, W_n\}$. Let $\mathcal{T} = \{T_1, \ldots, T_m\}$ be a set of k-length record vectors, where for each $T_i \in \mathcal{T}$: (1) records of T_i are ordered according to scoring function $\mathcal{F}$, and (2) T_i is the top k answer for a

non empty set of possible worlds $W(T_i) \subseteq \mathcal{W}$. A *U-Topk* query, based on $\mathcal{F}$, returns $T^* \in \mathcal{T}$, where $T^* = argmax_{T_i \in \mathcal{T}}(\sum_{w \in W(T_i)} P(w))$.

- U-kRank retrieves k ordered records where the i-th record has the highest probability of ranking in the i-th position among all possible worlds. Let D be an uncertain data set with possible worlds space $\mathcal{W} = \{W_1, \ldots, W_n\}$. For $i = 1, \ldots, k$, let $\{x_i^1, \ldots, x_i^m\}$ be a set of records, where each record x_i^j appears at rank i in a non empty set of possible worlds $W(x_i^j) \in \mathcal{W}$ based on scoring function $\mathcal{F}$. A *U-kRanks* query, based on F, returns $\{x_i^* | i = 1, \ldots, k\}$, where $x_i^* = argmax_{x_i^j}(\sum_{w \in W(x_i^j)} P(w))$.

Following, a large amount of work has been dedicated to top k queries with different semantics.

Threshold based top k queries defined by Hua *et al.* [HPZL08b] aim to retrieve all records whose probability of being top k results in all possible worlds is no less than a given probability threshold. A probabilistic threshold top k query (PT-k) on an uncertain table T consists of a top k query Q and a probability threshold $p(0 < p \leq 1)$. For each possible world W, Q is applied and a set of k tuples $Q^k(W)$ is returned. For a tuple $t \in T$, the top k probability of t is the probability that t is in $Q^k(W)$ in all $W \in \mathcal{W}$, that is,

$$\Pr^k_{Q,T}(t) = \sum_{W \in \mathcal{W}, t \in Q^k(W)} \Pr(W)$$

When Q and T are clear from context, $\Pr^k_{Q,T}(t)$ is written as $\Pr^k(t)$ for the interest of simplicity. Then, the answer set to a PT-k query is the set of all tuples whose top k probability values are at least p, that is,

$$Answer(Q, p, T) = \{t | t \in T, \Pr^k_{Q,T}(t) \geq p\}.$$

Then, techniques to efficiently compute the answer set for a PT-k query on an uncertain table were proposed in [HPZL08b].

Global-Top$k$ semantics in probabilistic relations was proposed in [ZC08], which return k highest-ranked tuples according to their probability of being in the top k answers in possible worlds. The *Global-Topk Probability* and *Global-Topk Answer Set over Probabilistic Relation* was defined.

- Global-Top$k$ Probability: Assume a probabilistic relation $R^p = \langle R, p, C\rangle$, a non-negative integer k and an injective scoring function s over R^p. For any tuple t in R, the Global-Top$k$ probability of t, denoted by $P_{k,s}^{R^p}(t)$, is the sum of the probabilities of all possible worlds of R^p whose top k answer contains t.

$$P_{k,s}^{R^p}(t) = \sum_{W \in pwd(R^p), t \in top_{k,s}(W)} \Pr(W).$$

- Global-Top$k$ Answer Set over Probabilistic Relation: Given a probabilistic relation $R^p = \langle R, p, C\rangle$, a non-negative integer k and an injective scoring function s over R^p, a top k answer in R^p under s is a set T of tuples such that

  1. $T \subset R$;
  2. If $|R| < k$, $T = R$, otherwise $|T| = k$;
  3. $\forall t \in T$, $\forall t' \in R - T$, $P_{k,s}^{R^p}(t) \geq P_{k,s}^{R^p}(t')$.

Then, efficient algorithms for evaluating top k queries under the Global-Top$k$ semantics are proposed in [ZC08].

Expected rank is proposed to use for answering top k queries on uncertain data by Cormode *et al.* in [CLY09]. They define a set of properties for ranking tuples. These properties naturally hold on certain data but are missed by some top k definitions on uncertain data.

- Exact-k [ZC08]. The top k list should contain exactly k items. Let R_k be the set of tuples (associated with their ranks) in the top k query result. If $|\mathcal{D}| \geq k$, then $|R_k| = k$.

- Containment. The top $(k+1)$ list should contain all items in the top k. For any k, $R_k \subset R_{k+1}$.

- Unique ranking. Within the top k, each reported item should be assigned exactly one position: the same item should not be listed multiple times within the top k. Let $r_k(i)$ be the identity of the tuple from the input assigned rank i in the output of the ranking procedure. The unique ranking property requires that $\forall i \neq j, r_k(i) \neq r_k(j)$.

- Value invariance. The scores only determine the relative behavior of the tuples: changing the score values without altering the relative ordering should not change the top k. Let D denote the relation which includes score values $v_1 \leq v_2 \leq \ldots$. Let s'_i be any set of score values satisfying $v'_1 \leq v'_2 \leq \ldots$, and define $\mathcal{D}'$ to be $\mathcal{D}$ with all scores v_i replaced with v'_i. The *value invariance* property requires that $R_k(\mathcal{D}) = R_k(\mathcal{D}')$ for any k.

- Stability [ZC08]. Making an item in the top k list more likely or more important should not remove it from the list. In the tuple-level model, given a tuple $t_i = (v_i, p(t_i))$ from $\mathcal{D}$, if we replace t_i with $t_i^{\uparrow} = (v_i^{\uparrow}, p(t_i^{\uparrow}))$ where $v_i^{\uparrow} \geq v_i$, $p(t_i^{\uparrow}) \geq p(t_i)$, then

$$t_i \in R_k(\mathcal{D}) \Rightarrow t_i^{\uparrow} \in R_k(\mathcal{D}'),$$

  where $\mathcal{D}'$ is obtained by replacing t_i with $t_i^{\uparrow}$ in $\mathcal{D}$. For the attribute-level model, the statement for stability remains the same but with $t_i^{\uparrow}$ defined as follows. Given a tuple t_i whose score is a random variable X_i, we obtain $t_i^{\uparrow}$

by replacing X_i with a random variable $X_i^{\uparrow}$ that is stochastically greater or equal than [SS94] X_i, denoted as $X_i^{\uparrow} \succeq X_i$.

They initially indicate a simple approach to get top k objects is to just compute the expected score of each tuple, and rank by this score, then take the top k. However, this is very dependent on the values of the scores: consider a tuple which has very low probability but a score that is orders of magnitude higher than othersthen it gets propelled to the top of the ranking, since it has the highest expected score, even though it is unlikely. But if this score is reduced to being just greater than the next highest score, the tuple will drop down the ranking. It therefore violates value invariance and ignores all the correlation rules completely in the tuple-level mode. Therefore, they study the score distribution, and then propose expected rank ranking method to typical answer top k queries on uncertain data:

Definition 2.1. *The rank of a tuple t_i in a possible world W is defined to be the number of tuples whose score is higher than t_i (so the top tuple has rank 0), i.e.,*

$$rank_W(t_i) = |\{t_j \in W | v_j > v_i\}|$$

By presenting the score distribution of all top k vectors, the users are able to choose among all results along the score-probability dimensions. Instead of displaying distributions of all potential top k vectors, the authors also propose to provide a number of typical vectors that effectively sample this distribution.

Finally, Li *et al.* [LSD09, LSD11] propose a unified approach to top k ranking in uncertain databases, based on parameterized ranking functions (PRFs), which could generalize many of the previously proposed ranking functions by setting weight function:

Definition 2.2. *Let $\omega : T \times \mathbb{N} \to \mathbb{C}$ be a weight function that maps a tuple-rank pair to a complex number. The parameterized ranking function (PRF), $\Upsilon_\omega : T \to \mathbb{C}$*

in its most general form is defined to be:

$$\begin{aligned}\Upsilon_\omega(t) &= \sum_{pw:t\in pw} \omega(t, r_{pw}(t)) \cdot Pr(pw) \\ &= \sum_{pw:t\in pw} \sum_{i>0} \omega(t,i) Pr(pw \wedge r_{pw}(t) = i) \\ &= \sum_{i>0} w(t,i) \cdot Pr(r(t) = i)\end{aligned}$$

A top k query returns the k tuples with the highest $|\Upsilon_\omega|$ values.

Then, depending on the actual function ω, different ranking functions with diverse behaviors can be generated, such as:

- Ranking by probabilities: If $\omega(t,i) = 1$, the result is the set of k tuples with the highest probabilities [RDS07].

- Expected score: By setting $\omega(t,i) = \text{score}(t)$, we get the E-Score:

$$\begin{aligned}\Upsilon_\omega(t) &= \sum_{pw:t\in pw} \text{score}(t)\text{Pr}(pw) \\ &= \text{score}(t)\text{Pr}(t) \\ &= \text{E}[\text{score}(t)]\end{aligned}$$

- Probabilistic threshold Top k (PT(h)): If we choose $\omega(i) = \delta(i \leq h)$, *i.e.*,

$$\omega(i) = \begin{cases} 1, for\ i \leq h \\ 0, otherwise \end{cases}$$

  , then the answer for PT(h) is gotten exactly.

- Uncertain rank-k (U-Rank): Let $\omega_j(i) = \delta(i = j)$, for some $1 \leq j \leq k$. We can see the tuple with largest Υ_{ω_j} value is the rank-j answer in U-Rank query [SIC07]. This allows us to compute the U-Rank answer by evaluating Υ_{ω_j} for all $t \in T$ and $j = 1, \ldots, k$.

- Expected ranks (E-Rank): Let PRF^l (PRF linear) be another special case of the PRF^ω function, where $w_i = \omega(i) = -i$. The PRF^l function bears a close similarity to the notion of expected ranks. Recall that the expected rank of a tuple t is defined to be:

$$\text{E}[r_{pw}(t)] = \sum_{pw \in PW} \Pr(pw) r_{pw}(t)$$

where $r_p w(t) = |pw|$ if $t_i \notin pw$. Let C denote the expected size of a possible world. It is easy to see that: $C = \sum_{i=1}^{n} p_i$ due to linearity of expectation. Then the expected rank of t can be seen to consist of two parts:

1. the contribution of possible worlds where t exists:

$$er_1(t) = \sum_{i>0} i \times \Pr(r(t) = i) = -\Upsilon(t)$$

where $\Upsilon(t)$ is the PRF^l value of tuple t.

2. the contribution of worlds where t does not exist:

$$\begin{aligned} er_2(t) &= \sum_{pw: t \notin pw} \Pr(pw)|pw| \\ &= (1 - p(t))(\sum_{t_i \neq t} \Pr(t_i | t \textit{ does not exist})) \end{aligned}$$

If the tuples are independent of each other, the following can be gotten:

$$\sum_{t_i \neq t} \Pr(t_i | t \textit{ does not exist}) = (C = p(t))$$

Thus, the expected ranks can be computed.

They propose a polynomial time algorithm based on generating functions to compute PRF. Consider the following generating function:

$$\begin{aligned} \mathcal{F}^i(x) &= (\prod_{t \in T_{i-1}} (1 - \Pr(t) + \Pr(t) \cdot x)) \Pr(t_i) \cdot x \\ &= \sum_{j \geq 0} c_j x^j \end{aligned}$$

Therefore, the coefficient c_j of x^j in the expansion of $\mathcal{F}^i$ is exactly the probability that t_i is at rank j, i.e., $c_j = \Pr(r(t_i) = j)$, and $\mathcal{F}^i$ contains at most $i+1$ non-zero terms. This both can be observed from the form of $\mathcal{F}^i$ above and also from the fact that $\Pr(r(t_i) = j) = 0$ if $j > i$. Hence, $\mathcal{F}^i$ can be expanded to compute the coefficients in $O(i^2)$ time. This allows us to compute $\Pr(r(t_i) = j)$ for t_i in $O(i^2)$ time; $\Upsilon(t_i)$, in turn, can be written as follows:

$$\Upsilon(t_i) = \sum_j \omega(t_i, j) \cdot \Pr(r(t_i) = j) = \sum_j \omega(t_i, j) c_j$$

which can be computed in $O(i^2)$ time.

2.2.2 Nearest Neighbor Queries

Nearest neighbor (NN) query over uncertain data is one of the most flourishing topic. Cheng *et al.* [CPK03, CKP04] are the first to tackle the probabilistic nearest neighbor (PNN) query, whose aim is to determine probabilistic candidates for the nearest neighbor of a given target along with corresponding probability values. In [KKR07], Kriegel *et al.* show nearest neighbor queries are an important query type for commonly used feature databases. Cheng *et al.* [CKP03] propose the Constrained Nearest Neighbor Query (C-PNN), which returns the IDs of objects whose probabilities are higher than some threshold, with a given error bound in the answers. The C-PNN can be answered efficiently with probabilistic verifiers. These are methods that derive the lower and upper bounds of answer probabilities, so that an object can be quickly decided on whether it should be included in the answer. Then they have developed three probabilistic verifiers, which can be used on uncertain data with arbitrary probability density functions. Only objects with probability no less than this threshold of being nearest neighbor of the query object will be output.

A KNN query has various applications in spatial databases. Consider, for example, a set of points in two dimensions representing cities. A KNN query may be issued to find "what are the k closest cities from a point p" Zhang *et al.* [ZLZ+10c] employ a rank based approach to process probabilistic KNN query, where k closest objects are returned according to their expected ranks.

It is important that answer probabilistic reverse nearest neighbor queries have been proposed in [LC09a]. In their approach, they approximate the uncertain objects by circular regions whereas we approximate the uncertain objects by rectangular regions. Based on these circular regions, they propose some pruning techniques to shortlist a set of candidate objects. At around the same time, Cheema *et al.* [CLW+10] formalize PRNN query that is to retrieve the objects from the uncertain data that have higher probability than a given threshold to be the RNN of an uncertain query object. Recently, Bernecker *et al.* [BEK+11] propose a solution to answer RKNN queries on uncertain data. They approximate the uncertain objects by rectangular regions, propose new pruning rules, and present the techniques to answer RKNN queries for $k > 1$. Their experimental results demonstrate that their proposed approach is the most efficient approach till date.

Influential Facilities(Sites) Queries

Bichromatic reverse nearest neighbor query is first introduced by Korn *et al.* in [KM00]. Given a set $\mathcal{F}$ of facilities and a set $\mathcal{O}$ of objects, the influence of a facility F can be measured by the number of objects whose nearest neighbors is F. As one of its natural extension, the problem of finding top k influential facilities (TkIS), is proposed in [XZKD05]. Instead of computing the set of BRNNs for a given set of facilities, it returns the top k facilities with highest number BRNNs (influences). They provide novel pruning techniques based on a new metric called

$minExistDNN$, and found the top k most influential facilities by browsing trees once systematically. Furthermore, [HWQ+11] found k locations from a set of candidate locations with the largest influence values according to a set of customers. On the other hand, the problem of optimal-location is studied in [YWN11, WÖF+11], aiming to find optimal area or location to set up a new facility such that it can attract the greatest number of facilities.

In [ZHZZ12], Zheng *et al.* studies the problem of finding top k most influential facilities over uncertain objects. They assume that each object is characterized by multiple instances, and the facilities remain deterministic, and adopt the *expected rank* [CLY09] as the ranking function to define the order of the facilities with probabilistic influences. By viewing the influence of a facilities as its uncertain attribute, they naturally adopt the expected rank to define the uncertain TkIS query.

Given a set of facilities F, a set of uncertain objects U and a query region Q, all facilities inside Q are considered as the candidate facilities, i.e., $C_f = \{f | f \in f\}$, and all objects influenced by candidate facilities as the candidate objects $C_u = \{U | \exists f \in C_f, U \in PRNN(f)\}$. A possible world W is obtained by independently instantiating each uncertain object U to one of its possible instances u with the probability of $\Pr(u)$. Then, for an uncertain object, all the instances which are the reverse nearest neighbors of the same facility are equivalent with respect to computing the influences. Therefore,those instances can be considered as a group so that the total number of possible worlds will be reduced. For a particular possible world W, the rank of a facility $f \in C_f$ in W is defined to be the number of candidate sites whose influence is greater than s, i.e.,

$$r_W(f) = |\{f' \in C_f | I_W(f') > I_W(f)\}|$$

where $I_W(f)$ is the influence of facility s in possible world W. The expected rank

of a facility is the expectation of its ranks across all possible worlds, i.e.,

$$er(f) = \sum_{W \in \mathcal{W}} r_W(f) \times \Pr(W)$$

The smaller $er(f)$ is, the higher f ranks. The formal definition is that given a set of facilities S, a set of uncertain objects U, a query region Q, and a natural number k, the uncertain top k influential facility query returns the top k facilities in Q according to the *expected rank semantics* [CLY09].

Based on attribute-uncertain model, an uncertain object may be influenced by multiple facilities instead of one as the deterministic case, so they use PRNN search to get the probability mass function (pmf) of the influence of a facility, and then compute expected rank of a facility across all possible worlds. They propose a general filter-refine style approach which includes efficient PRNN search, effective pruning schemes and divide-and-conquer based refinement to obtain the query results.

2.2.3 Dominating Query

Top k dominating queries are introduced by Papadias *et al.* [PTFS05]. The query retrieves the k points that dominate the largest number of other points. It selects the most powerful objects by using the classical concept of dominance. Unlike skyline query, the result does not necessarily contain skyline points.

Top k dominating queries are widely studied on certain data. Yiu *et al.* [YM07] propose the CBT (cost-based traversal) algorithm to select the k points with the highest dominating *score* values. The algorithm CBT traverses an aR-tree level by level to calculate the bound score of the number of points dominated by a point in an entry of aR-tree to prune the entry with lowest score. It shows the best overall performance over aggregate R-tree. Yiu *et al.* also give an extensive study on

the evaluation of top k dominating queries in [YM09]. They first propose a set of algorithms that apply on indexed data, and then they investigate query evaluation on data that are not indexed. They consider the dominance relationships between points in all dimensional subspaces. In [TPM11], the authors study progressive algorithms for top k dominating queries, where the user expresses an interest in a subset of the available dimensions.

Recently, uncertain query processing has received an increasing attention in many applications. Top k dominating query on uncertain data has been studied in [LC09b, LC13]. They propose probabilistic top k dominating (PTD) query to retrieve k uncertain objects that are expected to dynamically dominate the largest number of uncertain objects. Zhang *et al.* [ZLZ+10a] formalize threshold-based probabilistic top k dominating queries to overcome some inherent computational deficiency in an exact computation. Nevertheless, none of them supports the *parameterized ranking* semantics [LSD09, LSD11]. Moreover, as shown in [CLY09] the top k semantics adopted in [LC13, LC09b, ZLZ+10a] cannot properly capture the ranking of both probabilities and values. Recently, Feng *et al.* investigate the problem of probabilistic top k dominating query over sliding windows [FZGZ13]. However, they only consider the x-tuple uncertain model where each uncertain object is represented by one instance with a particular appearance probability. Techniques proposed in [LC09b, LC13, ZLZ+10a, FZGZ13] cannot be applied to the top k dominating model proposed in Chapter 4 due to the inherent difference of the models.

2.2.4 Skyline Operator

Börzsönyi *et al.* [BKS01] first investigate the skyline operator in the context of databases and propose an SQL syntax for the skyline query. The problem of spatial

skyline is first proposed in [SS06]. Given a set $\mathcal{O}$ of objects and a set $\mathcal{Q}$ of query points, each object has $|\mathcal{Q}|$ derived spatial attributes, each of which is the distance of the object to a query point in $\mathcal{Q}$, and hence can be mapped to a point in $|\mathcal{Q}|$-dimensional space where $|\mathcal{Q}|$ is the number of query points in Q. Then the spatial skyline regarding $\mathcal{O}$ and $\mathcal{Q}$ is the traditional skyline on $|\mathcal{Q}|$-dimensional space.

Skyline analysis is very useful in multi-criteria decision making applications. The work in [PJLY07] provides a first approach to probabilistic skyline computation. As an example, consider analyzing NBA players using multiple technical statistics criteria. Ideally, we want to find the perfect player who can achieve the best performance in all aspects. Unfortunately, such a player does not exist. The skyline analysis here is meaningful since it discloses the trade-off between the merits of multiple aspects. A player U is in the skyline if there exists no another player V such that V is better than U in one aspect, and is not worst than U in all other aspects. Skyline analysis on the technical statistics data of NBA players can identify excellent players and their outstanding merits.

The concept of probabilistic skyline was proposed in [PJLY07], which address two major challenges about skyline analysis and computation on uncertain data.

- Modeling Skylines on Uncertain Data: They introduce the probabilistic nature of uncertain objects into the skyline analysis. They calculate the probability that one object dominates the other to compare the advantages between two objects. Based on the probabilistic dominance relation, they propose the notion of probabilistic skyline. The probability of an object being in the skyline is the probability that the object is not dominated by any other objects.

- Efficient Computation of Probabilistic Skylines: They develop two algorithms to tackle the problem. The bottom-up algorithm computes the skyline probabilities of some selected instances of uncertain objects, and uses those in-

stances to prune other instances and uncertain objects effectively. The top-down algorithm recursively partitions the instances of uncertain objects into subsets, and prunes subsets and objects aggressively. The methods are efficient and scalable. As verified by the extensive experimental results, the methods are tens of times faster than the straightforward method.

The definition of the skyline operator is given in [LYZZ07]. For a given data set, the skyline operator returns all points in the data set which are not dominated by other points. Therefore, all players that lie on the skyline may be considered outstanding players. Most skyline analysis only use certain data in the form of the mean performance of the different players. Given a set of d-dimensional points, the skyline consists of the points, called "skyline points", which are not dominated by another point. A point $p = (p[1], p[2], \ldots, p[d])$ *dominates* another point $q = (q[1], q[2], \ldots, q[d])$ iff $p[i] \leq q[i]$ (for $1 \leq i \leq d$) and there is at least one dimension j such that $p[j] < q[j]$. The skyline computation (or the skyline operator) is crucial to many multi-criteria decision making applications. A typical example is a list of hotels, each of which contains two numerical attributes *distance* (say, to the beach) and *price*, for on-line booking. Figure 2.1 shows a sample list. In this application, the best choice to a client, who wants to spend holiday in the beach, may be as close as possible to the beach while also cost effective. Consequently, the "best" choices form the skyline are given in Figure 2.2.

However, the skyline operator and top k dominating queries rank objects in different ways: skyline ranks objects in a "defensive" way and outputs the objects which are not worse than any other objects in a given dataset, while a top k dominating query ranks objects in an "assertive" way and provides the objects that are better than the largest number of other objects. As pointed out in [YM09], the benefit of using top k dominating queries is to assimilate the advantages of top

id	*dist* (km)	*price* ($)
p_1	4	150
p_2	3	110
p_3	2.5	240
p_4	2	180
p_5	1.7	270
p_6	1	195
p_7	1.2	210

Figure 2.1: Hotel Information

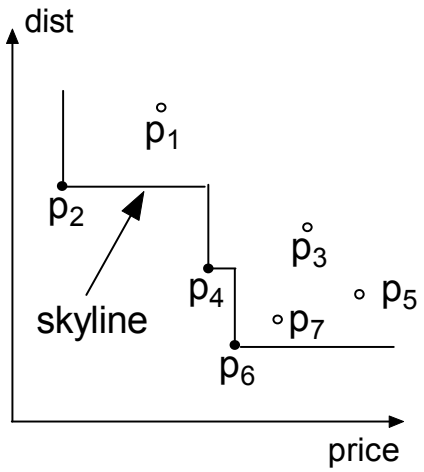


Figure 2.2: Skyline

k queries and the skyline operator. That is, the result size in a top k dominating query is strictly controlled by k.

2.2.5 Range Query

The aim of range queries is to find all the objects in a given range. Because the objects are uncertain, we cannot know their exact positions and their membership in the range. As shown in [TCX+05, TXC07], the range query may be uncertain in some applications. For instance, in the location based service a query point (e.g., a mobile device) may be represented by an uncertain object Q due to the inaccuracy of the measurement. Then the *uncertain range query* is represented by an uncertain object Q, a query distance γ and a probabilistic threshold θ; that is, find the uncertain objects whose *appearance probabilities*, denoted by $P_{app}(U, Q, \gamma)$, regarding Q and γ are not smaller than θ. Note that the distance between two points x and y is denoted by $\delta(x, y)$. For *continuous* case, we have $P_{app}(U, Q, \gamma) = \int_{y \in Q_r} \int_{x \in U_r \Lambda \delta(x,y) \leq \gamma} Q.pdf(y) \times U.pdf(x) dxdy$. As to the *discrete* case, $P_{app}(U, Q, \gamma) = \sum_{q \in Q \Lambda \delta(q,u) \leq \gamma} \sum_{u \in U} q_p \times u_p$. The formal definition of uncer-

tain range search is as follows.

Definition 2.3 (Uncertain Rang Search)**.** *Given a set $\mathcal{U}$ of uncertain objects, an uncertain range query Q, a distance γ and a probabilistic threshold θ ($0 < \theta \leq 1$), the uncertain range search retrieves the objects $U \in \mathcal{U}$ such that $P_{app}(U, Q, \gamma) \geq \theta$.*

U-tree [TCX+05, TXC07] is the first index structure supporting range queries on multidimensional spatial uncertain data with arbitrary PDFs. U-tree is a new modification of R-tree by using a set of new pruning and validating techniques. The techniques [KKPR06, SMP+07, ZLZ+10b, AF12] are also proposed to do range search against multi-dimensional uncertain data with arbitrary PDF. These techniques follow the *filtering-and-verification* paradigm such that, by taking advantage of the indexing structures, many objects are *filtered* at a reasonable filtering cost without explicit calculation of their *appearance probabilities*. The essential idea of the existing techniques is as follows: a summary of the PDF is pre-computed for each uncertain object to approximately capture the distribution of its PDF, and the summaries of the uncertain objects are organized by augmenting existing index techniques (e.g., R-tree). For a given search region r_q, the lower and upper bounds of the *appearance probability* can be derived at a cheap cost for each uncertain object. Then an uncertain object U may be *filtered* in two ways:

1. U is pruned if the upper bound of the *appearance probability* is smaller than the given probabilistic threshold θ.

2. U is validated (*qualified*) if the lower bound of the *appearance probability* is not less than θ. Only the objects survived the filtering phase need to be verified, i.e., explicitly computing their *appearance probabilities*.

Recently, Zhang *et al.* propose a novel technique called U-Quadtree to effectively index the multi-dimensional uncertain objects with arbitrary PDFs to support

range search in [ZZLL12, ZZL+14]. All the index structures will be introduced in Section 2.3.

There are also some studies on indexing multi-dimensional uncertain objects which focus on specific cases of objects' PDFs and queries. For instances, in [BPS06, BGK+07], Böhm *et al.* study range queries with the constraint that PDFs of uncertain objects follow *Gaussian* distributions. Assuming PDFs of the objects are either histograms or more complex ones such as *Gaussian* or piecewise algebraic. In [ACTY09], Agarwal *et al.* provide thorough theoretical analysis on range search on uncertain data. Managing uncertain moving objects [ZCJ+09] and uncertain categorical data [SMP+07] have been separately studied. Aggarwal *et al.* [AY08] study the problem of indexing high dimensional uncertain data with the assumption that the PDFs of the uncertain object on each dimension are independent to others. Assuming the space is partitioned by a *virtual grid* with limited number of cells, Ma *et al.* [MKM08] propose solutions for efficient retrieval of uncertain spatial point data where the location information is derived from the free text by *spatial expressions*. Recently, Kinura *et al.* [KMZ10] propose a primary indexing technique named UPI to speed up the query processing on uncertain data by clustering the heap files, in which U-tree [TCX+05, TXC07] technique is used as a building block to index uncertain objects with arbitrary PDFs. In [LC10], Lian *et al.* propose a generic framework to index uncertain data. Their main focus is how to tackle the local correlations among uncertain objects, and their indexing technique falls in the R-tree category.

2.2.6 Similarity Join

An important database primitive for commonly used feature databases is the similarity join. It combines two datasets based on some similarity predicate into one

set such that the new set contains pairs of objects of the two original sets. In many different application areas, e.g. sensor databases, location based services or face recognition systems, distances between objects have to be computed based on vague and uncertain data. Conventional join queries over two multi-dimensional datasets are fundamental in data analysis and information retrieval. Most existing techniques for join queries have been developed based on popular spatial access methods such as R-trees. For threshold based joins, there are three main stream spatial join algorithms using R*-tree [HKL$^+$09]. They are the depth-first-join (DFJ) algorithm [BKS93], the breadth-first-join (BFJ) algorithm [HJR97], and transformation-view-join (TVJ) algorithm [LWHS06]. Techniques for top k spatial/similarity queries are studied in [CMTV00, HS98]. Various algorithms, such as exhaustive algorithm, recursive algorithm, Heap algorithm, and priority queue based algorithms are proposed. Many variation of join queries over multi-dimensional space have been studied in different contexts, including road networks [SAS06], moving objects [ZLRB08] and data streams [SCL$^+$12].

Join queries over uncertain objects are inherently different than conventional joins where each uncertain object takes a set of mutually exclusive instances/points in a multi-dimensional space. The existing model for handling similarity joins over objects with multiple instances follows the probabilistic semantics on uncertain objects [CSP$^+$06, KKPR06, LS08] and aims to capture relative instance distribution among objects with multiple instances. Note that all instances in a multi-valued object exist simultaneously instead of mutually exclusive in an uncertain object. Due to such inherent differences in semantics, join techniques over uncertain objects cannot be directly applied to similarity joins over multi-valued objects.

Top k similarity join, also called *closest pair queries*, has attracted much research attention [CMTV00]. In many applications such as decision making and

e-business, an object may be represented by multiple points (instances) in the d-dimensional space, namely *multi-valued* objects [EN11].

Definition 2.4 (Top k Similarity Join)**.** *Given two sets of objects (points) $\mathcal{U}$ and $\mathcal{V}$ in a d-dimensional metric space, the top k similarity join query retrieves k pairs of objects $\mathcal{P}$ from $\mathcal{U} \times \mathcal{V}$ such that the distance between any pair of objects in $\mathcal{P}$ is not greater than the distance of any object pairs in $\mathcal{U} \times \mathcal{V} - \mathcal{P}$.*

Uncertain objects are inherently different than multi-valued objects. Instances of an uncertain object are mutually exclusive which means at most one instance can appear at a particular time, while all the values/instances of a multi-valued object must occur simultaneously at any time. Moreover, as shown in [ZLC+10], models based on uncertain semantics cannot always capture the relative distributions of multi-valued objects. In [ZXL+12], Zhang *et al.* investigate the top k similarity join problem over multi-valued objects based on a ϕ-quantile distance ($\phi \in (0, 1]$).

2.3 Indexing Uncertain Objects

Recently, there are several index techniques proposed to speed up queries on uncertain data. We will give an overview of them below.

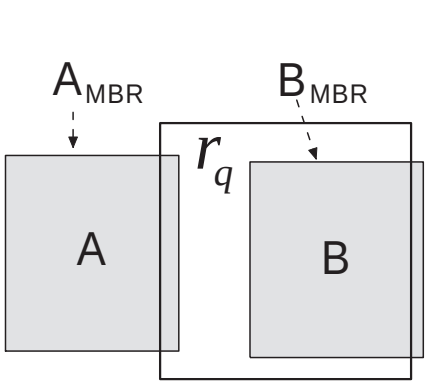


Figure 2.3: MBR

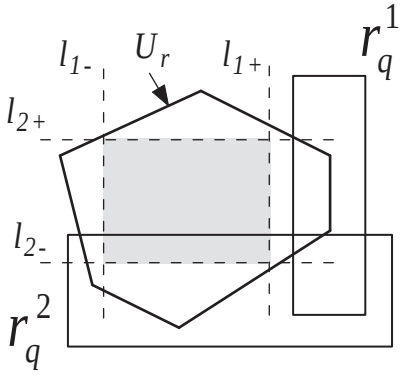


Figure 2.4: PCR

***R*-tree.** In *R*-tree approach [KKPR06, SMP+07, LC10], the MBR of an object serves as the summary of its PDF, and MBRs are organized by *R*-tree. An uncertain object U can be validated if its MBR is contained by r_q, i.e., $U_{mbr} \subseteq r_q$, regardless of the value of the probabilistic threshold θ. Similarly, U is pruned if U_{mbr} does not intersect r_q, i.e., $U_{mbr} \cap r_q = \emptyset$. This approach is simple and performs well if the uncertain region sizes are much smaller than r_q. However, as the MBR cannot further explore the PDF of an uncertain object, the filtering capacity of the index is poor when the size of the uncertain region is not small. As shown in Figure 2.3, for a given search region r_q, we cannot prune A regarding any probabilistic threshold θ although intuitively $P_{app}(A, r_q)$ should be small. Similarly, B cannot be validated either. Moreover, Papadias *et al.* propose aggregate *R*-tree to index the instance of each uncertain object [PKZT01], which is called Local aR-tree. Then, the previous *R*-tree indexing the object MBRs is called Global *R*-tree.

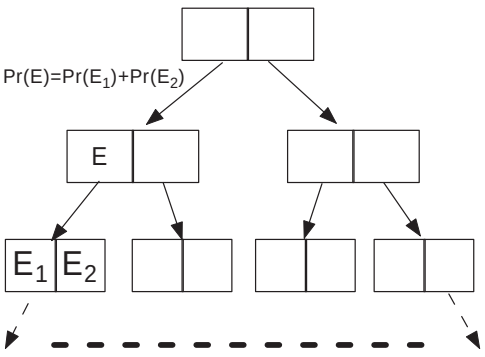


Figure 2.5: Local aR-tree

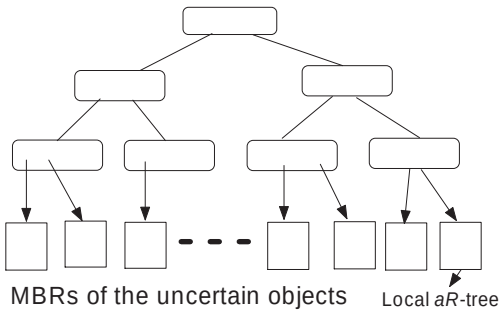


Figure 2.6: Global *R*-tree

- Local aR-tree. For each uncertain object, a *local* aR-tree is built to organize its instances. The aggregate information kept on each intermediate entry is the sum of weights of instances indexed by the entry. Namely, for every intermediate entry E in the local aR-tree, the probability of E is recorded as the sum of probability of instances having E as an ancestor. For example, in

Figure 2.5, the entry E is the parent of the entries (leaves) E_1 and E_2, so the probability of E equal to the sum of probability of E_1 and E_2.

- Global R-tree. As showed in Figure 2.6, for each object in $\mathcal{U}$, we first obtain the MBR of its instances. Then we build an R-tree on these MBRs. This R-tree is called the *global* R-tree of $\mathcal{U}$, which indexes the MBRs of all uncertain objects. In a global R-tree, each leaf (data) entry is an MBR of an uncertain object.

U-tree. The PDF summary of an object in U-tree is a finite set of probabilistically constrained regions (PCRs), which is introduced by Tao *et al.* in [TCX+05, TXC07]. PCR is a general version of *x-bounds* which aims to index one dimensional uncertain data [CXP+04]. For a given θ ($0 \leq \theta \leq 0.5$), the PCR of an object U regarding θ, denoted by $U.pcr(\theta)$, is constructed as follows. As shown in Figure 2.4, in each dimension, two lines are calculated. In the horizontal dimension, U has the probability θ to occur on the left side of line l_{1-}, also probability θ to occur on the right side of line l_{1+}. Similarly, l_{2-} and l_{2+} are calculated in the vertical dimension. The shaded region in Figure 2.4 is the geometric representation of $U.pcr(\theta)$. Then we can take advantage of $U.pcr(\theta)$ to prune or validate U regarding θ and r_q. For instance, as shown in Figure 2.4, suppose θ is the probabilistic threshold for two search regions r_q^1 and r_q^2. U can be pruned regarding r_q^1 because r_q^1 does not intersect $U.pcr(\theta)$. On the other hand, U can be validated with respect to r_q^2 since all instances below l_{2-} are contained by r_q^2. As it is infeasible to keep all $U.pcr(\theta)$ for any $\theta \in [0, 0.5]$, a finite number of PCRs are pre-computed for each object and the lower and upper *appearance probability* bounds can be derived. Based on the PCRs of the uncertain objects, U-tree is built up in a similar way with R-tree where each entry in a leaf node corresponds to the PCRs of an uncertain object.

UI-tree. The PDF summary of an object U in [ZLZ+10b] is a set of groups which are disjointed partitions of its PDF based on a KD-Tree. Given a search region r_q, we can derive the lower and upper bounds of $P_{app}(U, r_q)$ based on the topological relationships between the groups and r_q. Specifically, groups contained by r_q contribute to both lower and upper bounds of the *appearance probability* since all instances in these groups are contained by r_q. With similar rationale, groups overlapped by r_q only contribute to upper bound. Then an object U may be validated (pruned) based on the lower (upper) bound of $P_{app}(U, r_q)$. For the space efficiency, the groups of the uncertain objects may be merged such that a set of groups from different objects can share the same boundary, namely "word" in [ZLZ+10b]. The identifications of the related objects and their corresponding probability mass are kept in each "word". Then UI-tree is constructed in a similarly way with R-tree where each entry of a leaf node is a "word".

UP-Index. Recently, Angiulli *et al.* [AF12] develop a pivot based indexing mechanism for uncertain data in general metric space. For a given pivot point p and an object U, the PDF summary of U is the histogram of the distance distribution regarding p and U. The upper bound of $P_{app}(U, r_q)$ can be derived based on the *reverse triangle inequality* according to the histogram and the distance between the center of r_q and the pivot point p. Then an object can be pruned based on the upper bound derived. To enhance the pruning power, a set of pivot points are employed in [AF12]. The advantage of UP-Index is that it can support distance based range query in general metric space. Nevertheless, as shown in our empirical study, its performance is not competitive under our problem setting because : (i) an object cannot be validated in [AF12] because UP-Index cannot derive the lower bound of $P_{app}(U, r_q)$, and (ii) for any range search the whole index is scanned to prune objects, and the index size is usually large for a decent pruning capacity.

***U*-Quadtree.** A quadtree [FB74] is a space partitioning tree data structure in which a d-dimensional space is recursively subdivided into 2^d regions (cells). In [ZZLL12, ZZL+14], the instances of an uncertain object U are organized by a *summary*, denoted by S_U, which consists of a set of entries $\{e\}$, where each entry records the object id ($e.oid$), the cell of the quadtree ($e.cid$) and the probability mass of instances allocated on this cell ($e.p$). The entries of the uncertain objects are organized by a $B+$ tree where the cell ids are key values, which are generated based on Hilbert curve. The structure of U-Quadtree is introduced below.

Given a quadtree, a **summary** of an object U is defined as follows:

Definition 2.5 (S_U)**.** *A **summary** S_U of an object U regarding a quadtree consists of a set of entries $\{e\}$ where each entry e is a triplet ($e.c$, $e.o$, $e.p$) where $e.c$ and $e.o$ represent the identification (id) of the cell and the object associated with the entry, and $e.p$ ($0 < e.p \leq 1$) is the probability mass of the instances assigned to this entry (i.e., the cell $e.c$). For any instance $x \in U$, x must be assigned to exactly one cell c (entry e) where $x \in c$ ($e.c$) and hence $\sum_{e \in S_U} e.p = 1$.*

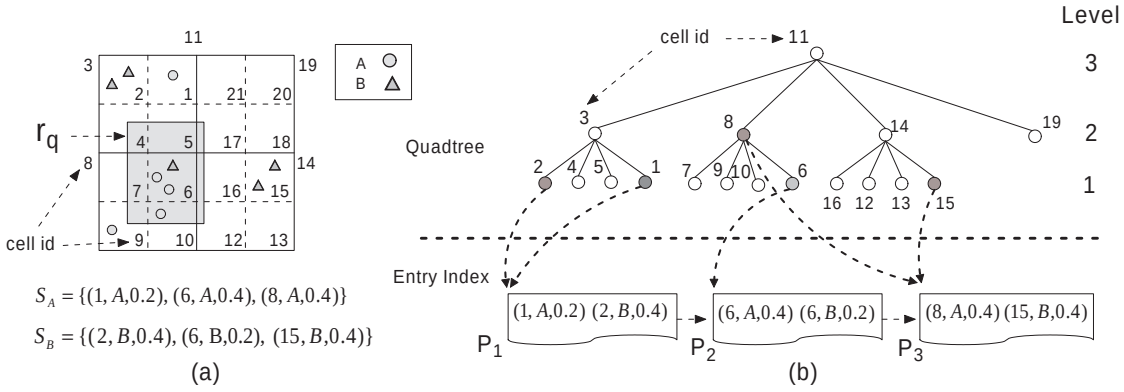


Figure 2.7: U-Quadtree

In Figure 2.7(a), the objects A and B have 5 instances each and all instances have the same occurrence probability (0.2). The height of the quadtree (h) is 3 and the ids of the cells are labeled. We may have S_A =

$\{(1, A, 0.2), (6, A, 0.4), (8, A, 0.4)\}$ and $S_B = \{(2, B, 0.4), (6, B, 0.2), (15, B, 0.4)\}$. Note that the summary of an object is not unique as an instance x can be assigned to any cell which *contains* x. For instance, an alternative of S_A could be $\{(1, A, 0.2), (8, A, 0.8)\}$ in which 4 instances of A are assigned to cell 8 on level 2.

As shown in Figure 2.7(b), entries of the objects are organized based on a quadtree, named U-Quadtree, which consists of two parts:

- **Entry Index** (UQ_E): a B^+ tree used to keep entries of the objects in the secondary memory, where the key of each entry is its cell id. Similar to [HS02], we assume the id of a cell is its Hilbert code [Fal88] generated in a recursive way such that the cells with close spatial proximity are likely to be allocated to the same or adjacent pages in UQ_E. Particularly, a leaf node of UQ_E is called the *entry page* (e.g., P_1, P_2 and P_3 in Figure 2.7(b)) and f denotes its capacity (i.e., the maximal number of entries in an entry page).

- **Quadtree** (UQ_T): a pointer-based quadtree with height h. For each cell (node) c, let P be the first entry page in which there is an entry e with $e.c = c$. We keep the address of P as the pointer of cell c. As shown in Figure 2.7(b), a gray cell (node) of UQ_T implies the pointer of the cell is not empty, i.e., there is at least one entry on it. Note that we do not need to keep a cell in UQ_T if none of its descendent cells including itself contains any entry.

2.4 Uncertain Trajectories

Recent years have witnessed the increasing amount of research on uncertain data modeling and query processing due to their importance in many applications. In

this subsection, we briefly introduce two categories of work closely related to the problem studied in this chapter.

2.4.1 Uncertain Trajectories Modeling.

A variety of models have been proposed to capture the uncertainty of the trajectory data. Early studies on uncertain trajectories employ simple geometric shapes (e.g., cylinders [TWHC04] and beads [PJ99]) to approximate the possible locations of a moving object. Despite of its simplicity, this model suffers from an inherent drawback: the probability distribution of an object is not considered and hence cannot appropriately support probabilistic queries. In some recent work (e.g., [ZTZS11]), the network-constraint model is used where the raw location of a moving object is mapped to a linear range on the road networks. In [CKP04, MLSC13, XYCL13], the uncertain location of an object is captured by an independent probability density function (e.g., Gaussian distribution) at each point of time. As shown in [EKM$^+$12b, NZE$^+$13], the temporal dependence between two subsequent locations of an object is lost in this model. Recently, a novel evolving density model is proposed in [JLSY14] to capture the time-varying uncertainty of the moving objects where the probability density function may change over time. Markov Chain model has been widely used in the literature to capture the temporal dependency of a moving object, and hence is naturally adapted to describe the uncertainty of the trajectory data with low sampling rate in [EKM$^+$12b, EKM$^+$12a, NZE$^+$13, XGC$^+$13]. Moreover, [EKM$^+$12b, EKM$^+$12a] show that the Markov Chain model correctly complies with the classical possible world semantics [DS07]. In our work, we employ Markov Chain model to describe the uncertain trajectories.

To model uncertain object trajectories, Emrich *et al.* [EKM$^+$12b] suppose that the locations of an uncertain spatio-temporal object $o \in \mathcal{D}$ at time t are realizations

of a random variable $o(t)$. An uncertain object trajectory of object $o \in \mathcal{D}$ comprises a set of trajectories, each assigned with a probability indicating its likelihood to be the true trajectory of o. This consideration suggests modeling uncertain object trajectories as a realization of a stochastic process [KT75], formally:

Definition 2.6 (Uncertain Object Trajectory)**.** *Given the spatial domain $\mathcal{S}$ and the time domain $\mathcal{T}$, an uncertain object trajectory $o(t) \in \mathcal{S}$ of an object $o \in \mathcal{D}$ is a stochastic process $\{o(t) \in \mathcal{S}; t \in \mathcal{T}\}$.*

2.4.2 Capture Uncertainty by Markov Chains

In [EKM+12b, EKM+12a], an uncertain trajectory is modeled as a realization of a stochastic process [KT75].

Markov Chains can model a discrete spatio-temporal (state-time) space with the assumption that $o(t+1)$ only depends on $o(t)$.

Definition 2.7 (Markov Chain model)**.** *Given a stochastic process $o(t)$ with $t \in \mathcal{T}$ and a state $s \in \mathcal{S}$, the stochastic process is called Markov Chain iff $P(o(t+1) = s_j | o(0) = s_0, o(1) = s_1, \ldots, o(t) = s_i) = P(o(t+1) = s_j | o(t) = s_i)$.*

For an object o moving on the space $\mathcal{S}$, we set $P_{i,j}(o) = P(o(t+1) = s_j | o(t) = s_i)$, where $P_{i,j}(o)$ represents the probability of object o moving from state s_i to s_j when the time changes from anytime time $t \in \mathcal{T}$ to its successive time $t+1$. We can store all $P_{i,j}(o)$ in a $n \times n$ matrix $M(o)$ to represent the transition probability of object o from state s_i to s_j at any time t, where the matrix $M(o)$ is called transition matrix. Then we have $o(t+1) = o(t) \times M(o)$. Recall that $o(t)$ is the distribution vector of an object o at time t where $\sum_{s \in \mathcal{S}} P(o(t), s) = 1$. Similarly, if $M(o)^T$ is defined as a transposed Markov Chain matrix, we have $o(t) = o(t+1) \times M(o)^T$.

Given two subsequent observations $o(t_i)$ and $o(t_j)$, efficient algorithm is proposed in [EKM+12b] to derive the location distribution $o(t)$ for $t \in (t_i, t_j)$ based on $M(o)$ and $M(o)^T$. Same as [EKM+12b, EKM+12a], we assume objects share the same Markov Chain matrix which can be learned by domain experts in various applications.

In an uncertain object trajectory of an object o, we have several consequent observation timestamps. The position of the object on each timestamp is certain, whereas the positions between two continuous timestamps are uncertain, which can be modeled by the two transition matrixes. For example, if we observe the object 0 on s_i when t_1 and then s_j when t_5, the positions of o on t_2, t_3, t_4 are unknown. Let $P(o, t_1) = (0|s_1, \ldots, 1|s_i, \ldots, 0|s_n)$, then the $P_f(o, t_2) = P(o, t_1) \times M(o)$, $P_f(o, t_3) = P(o, t_1) \times M(o)^2$, $P_f(o, t_4) = P(o, t_1) \times M(o)^3$. Similarly, Let $P(o, t_5) = (0|s_1, \ldots, 1|s_j, \ldots, 0|s_n)$, then the $P_b(o, t_4) = P(o, t_5) \times M(o)^T$, $P_b(o, t_3) = P(o, t_5) \times (M(o)^T)^2$, $P_b(o, t_4) = P(o, t_5) \times (M(o)^T)^3$. Note that P_f and P_b mean the result derived from $M(o)$ and $M(o)^T$ respectively. After combining the P_f and P_b, we can get the distributions of the object on t_2, t_3, t_4, and then the possible worlds are gotten.

2.4.3 Range Search on Uncertain Trajectories

Range search on uncertain data has been intensively studied in recent years. A large body of work (e.g., [TXC07, ZZLL12]) focus on the range search on a snapshot of uncertain trajectories; that is, each object is described by a probabilistic density function, and objects with appearance probability exceeding a given threshold are retrieved. The problem of range search on uncertain trajectories has been investigated against difference uncertain models such as cylinder model (e.g., [TWHC04]), beads model (e.g., [PJ99]), network-constraint model (e.g., [ZTZS11]), inde-

pendent probabilistic density function model (e.g., [CKP04]), evolving density model [JLSY14], as well as the Markov Chain model [EKM+12b, EKM+12a].

As to the best of our knowledge, [EKM+12b, EKM+12a] are only two existing work which study the problem of range search on uncertain trajectories modeled by Markov Chains. Particularly, Emrich *et al.* propose efficient computation algorithm for range search in [EKM+12b] without the support of indexing technique. In [EKM+12a], they further improve the performance by utilizing pre-computed sub-diamonds based summaries which can significantly reduce the number of candidate objects for refinement. Section 2.4.4 will introduce the sub-diamonds based filtering technique.

2.4.4 Sub-diamonds based Filtering

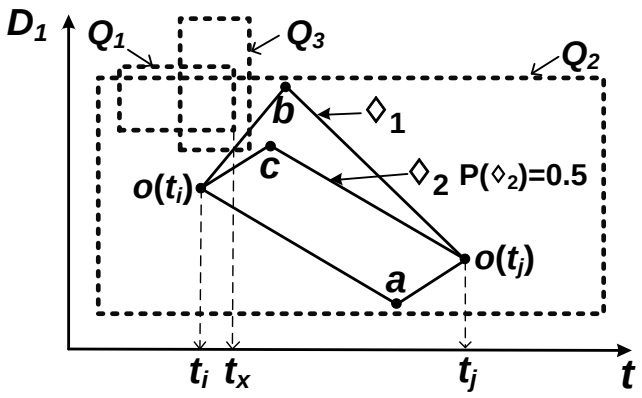


Figure 2.8: Sub-diamonds based Filtering

In this subsection, we briefly introduce the sub-diamonds based filtering technique, which is the key of the range search algorithm in [EKM+12a]. As shown in [EKM+12a], all possible valid trajectories within a segment $g(o, t_i, t_j)$ can be bounded by a diamond for each individual dimension if the maximal speed is given. Figure 2.8 depicts the diamond $\Diamond_1$ ($\Diamond_1 = \langle o(t_i), a, o(t_j), b\rangle$) where the horizontal axis represents the time and the vertical axis is the dimension D_1. Given a range

search query, we may easily *prune* the segment g based on $\Diamond_1$. For instance, we have $P(o(t), Q_1) = 0$ for time $t \in [t_i, t_x]$ since the query region Q_1 does not overlap $\Diamond_1$ regarding the dimension D_1. Thus, g can be excluded from further computation. Similarly, we can claim $o(t)$ is enclosed by Q_2 for any $t \in [t_i, t_j]$ if Q_2 contains the diamond regarding both D_1 and D_2.

Intuitively, the filtering performance can be further enhanced by maintaining a set of sub-diamonds. For instance, $\Diamond_2$ in Figure 2.8 is a sub-diamond of g where $\Diamond_2 = \langle o(t_i), a,\ o(t_j), c\rangle$, and we know that its associated probability, denoted by $P(\Diamond_2)$, is 0.5; that is, with probability at least 0.5, $o(t)$ is bounded by $\Diamond_2$ for any $t \in [t_i, t_j]$. Consequently, the segment g can be pruned for search region Q_3 if probability threshold $\theta \geq 0.5$ because Q_3 does not overlap $\Diamond_2$. The validation of the segment can be conducted in a similar way. Together with the diamonds and their minimal bounding rectangles, the sub-diamonds of the segments are organized by an R-tree in [EKM$^+$12a], namely UST-Tree.

Chapter 3

Find Top k Influential Facilities

We follow the popular *possible world semantics* to model the influence of each facility as a influence score distribution. The definition of *influence score* of a facility in each possible world is exactly the same as the traditional BRNN query since only one instance occurs for each uncertain object in a possible world. Then we apply the **maximal utility principle** to rank the facilities. The maximal utility principle [SS94] has been widely used in various applications such as economic, finance and mathematics, and it selects the one with **highest expected score** as the optimal solution among a set of score distributions. By assuming the uncertain regions of uncertain objects are organized by R-tree which is the most popular indexing technique used for uncertain objects in the literature, we propose an efficient algorithm following the *synchronized* R-tree traversal paradigm. Moreover, based on a recent uncertain indexing technique [ZZLL12], namely U-Quadtree, we further significantly improve the performance of the algorithm in terms of CPU and I/O costs. Finally, we approve the proposed algorithms can also apply to the samples of the uncertain objects with massive number of instances.

This chapter is organized as follows. We first formally define the problem of top

k most influential facilities over uncertain objects, and introduce some preliminary work in Section 3.1. In Section 3.2, we propose our exact algorithms based on R-tree and U-Quadtree respectively. Then Section 3.3 presents efficient randomized algorithms to provide approximate solutions with accuracy guarantee. Results of comprehensive performance studies are reported in Section 3.4. Finally, Section 3.5 concludes the chapter.

3.1 Background

We present problem definition and necessary preliminaries in this section. For references, notations frequently used in this chapter are summarized in Table 3.1.

Notation	Definition
U	an uncertain object
F	the facility or facility entry
u	instance of the uncertain object
n	number of uncertain objects
m	number of instances per uncertain object
I^+ (I^-)	upper(lower) bound of *expected score*
E	entry of R-tree
NND	Nearest Neighbor Distance
$nnd\_min(R_1, R_2)$	the minimal NND between rectangles R_1 and R_2
$nnd\_max(R_1, R_2)$	the maximal NND between rectangles R_1 and R_2
T, t	object tuple
$T.e$	entry associated with T
$T.\mathcal{F}$	the facilities which may influence the object associated with T
λ	pruning threshold for *expected score*
s	number of sample sets (rounds)
S_i	i-th sample set

Table 3.1: The summary of notations.

3.1.1 Problem Definition

A point (instance) x referred in this chapter, by default, is in a d-dimensional numerical space. Given two points x and y, the distance between them is denoted by $d(x, y)$. Euclidian distance metric is employed in this chapter, and the techniques developed in this chapter can be easily extended to other metric distances. In this chapter, we focus on the bichromatic nearest neighbor search. Given a set $\mathcal{F}$ of facilities (points) and the nearest neighbor of an object point x (x is not a facility) is its nearest facility, denoted by $NN(x)$; that is, $d(x, NN(x)) = \min\{d(x, F)|F \in \mathcal{F})\}$. Without loss of generality, we assume the nearest neighbor of a point x ($NN(x)$) is unique.

Uncertain Objects. An uncertain object can be described either *continuously* or *discretely*. In this chapter, we focus on *discrete case*. Note that we can discretize a continuous probability density function (PDF) of an uncertain object by sampling methods. In the *discrete cases*, an uncertain object consists of a set $\{u_1, u_2, \ldots, u_m\}$ of instances (points) where for $1 \leq i \leq m$, u_i occurs with the probability p_{u_i} ($p_{u_i} > 0$), and $\sum_{i=1}^{m} p_{u_i} = 1$. For an uncertain object U, U_{mbr} denotes the minimal bounding rectangle (MBR) of the instances of U.

Note that, in this chapter, we assume the facilities are represented by points because usually their locations can be obtained precisely.

The possible world semantics. Given a set of uncertain objects $\{U_1, U_2, \ldots, U_n\}$, a possible world $W = \{u_1, u_2, \ldots, u_n\}$ is a set of instances sequentially sampled from each object. Assume the uncertain objects are independent to each other, and the probability of W to appear is $Pr(W) = \prod_{i=1}^{n} p_{u_i}$. Let $\mathcal{W}$ denote the set of all possible worlds, then $\sum_{W \in \mathcal{W}} Pr(W) = 1$.

Example 3.1. *In Figure 3.1(a), $\mathcal{F}$ consists of three facilities F_1, F_2 and F_3, and there are three uncertain objects A, B and C. Both A and B have two instances*

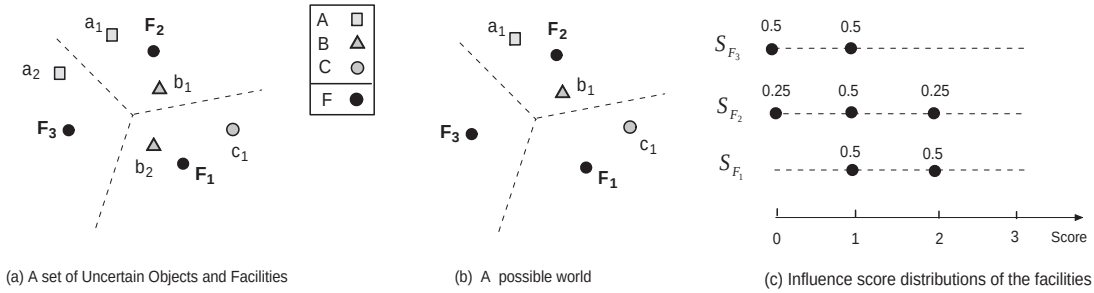


Figure 3.1: Example for the Problem Definition

with the same occurrence probability (0.5), while C has only one single instance c_1 *with* $p_{c_1} = 1.0$. *Consequently, there are totally* 4 *possible worlds in this example, where* $W_1 = \{a_1, b_1, c_1\}$, $W_2 = \{a_1, b_2, c_1\}$, $W_3 = \{a_2, b_1, c_1\}$, $W_4 = \{a_2, b_2, c_1\}$ *and the probability of each possible world is* 0.25. *Particularly, the possible world* W_1 *is illustrated in Figure 3.1(b).*

For each possible world W, let $s(F, W)$ denote the influence score of the facility F regarding W, which is the number of reverse nearest neighbors of F in W; In this chapter, for each facility F, we use S_F to represent the influence score distribution of F, where $Pr(S_F = v) = \sum_{W \in \mathcal{W} \wedge s(F,W)=v} Pr(W)$.

Example 3.2. *In Figure 3.1(b), we have* $s(F_1, W) = 1$, $s(F_2, W) = 2$ *and* $s(F_3, W) = 0$. *Figure 3.1(c) illustrates the score distributions of* F_1, F_2 *and* F_3. *For facility* F_1, *we have* $Pr(S_{F_1} = 1) = 0.5$, $Pr(S_{F_1} = 2) = 0.5$. *Similarly,* $Pr(S_{F_2} = 0) = 0.25$, $Pr(S_{F_2} = 1) = 0.5$, $Pr(S_{F_2} = 2) = 0.25$,$Pr(S_{F_3} = 0) = 0.5$ *and* $Pr(S_{F_3} = 1) = 0.5$.

The influence score distribution of a facility F (S_F) is a random variable, and hence we can apply the **maximal utility principle** to rank the facilities. The maximal utility principle [SS94] is one of the most popular models to select the one with **highest expected score** as the optimal solution among a set of score distributions. In this chapter, the *expected influence score* of a facility F, denoted

by $I(F)$, is defined as follows.

$$I(F) = \sum_{W \in \mathcal{W}} s(F, W) \times Pr(W) \tag{3.1}$$

As the number of possible worlds grows exponentially regarding the number of uncertain objects in $\mathcal{U}$ and the number of instances in each uncertain object, it is cost-inhibitive to apply Equation 3.1 straightforwardly by enumerating all possible worlds. Therefore, we will find an alternative of Equation 3.1 which can be derived with reasonable computational cost.

Let $NN(U, W)$ denote the nearest neighbor (facility) of U in the possible world W, and $\sigma(NN(U, W) = F) = 1$ if the facility F is the nearest neighbor of U in the possible world W, and $\sigma(NN(U, W) = F) = 0$ otherwise. Since we assume the uncertain objects are independent to each other, Equation 3.1 can be rewritten as follows.

$$\begin{aligned} I(F) &= \sum_{W \in \mathcal{W}} (\sum_{U \in \mathcal{U}} \sigma(NN(U, W) = F)) \times Pr(W) \\ &= \sum_{U \in \mathcal{U}} Pr(NN(U) = F) \\ &= \sum_{U \in \mathcal{U}} \sum_{u \in U \wedge NN(u) = F} p_u \end{aligned} \tag{3.2}$$

where $Pr(NN(U) = F)$ is the probability that F is the nearest neighbor of U, i.e., $Pr(NN(U) = F) = \sum_{u \in U \wedge NN(u) = F} p_u$ and $NN(u)$ is the nearest neighbor of the instance (point) u.

Equation 3.2 implies that we can avoid enumerating all possible worlds since we can independently compute $Pr(NN(U) = F)$ (i.e., nearest neighbor probability) for each uncertain object. Our empirical study shows that even a naive implementation of the Equation 3.2 can outperform the existing work which follows the *expected rank* model.

In the light of *maximal utility principle*, we aim to find the k facilities with highest *expected influence scores*, which is formally described below.

Problem Statement. Given a set of uncertain object $\mathcal{O}$ and a set of facility $\mathcal{F}$, find the k facilities with the highest *expected influence scores*. We assume the number of facilities $|\mathcal{F}| \geq k$, and ties are broken arbitrarily.

3.1.2 Expected Score vs Expected Rank

In [ZHZZ12], Zheng *et al.* propose the *expected rank* based ranking model to evaluate the influence of the uncertain objects. For a given facility F, its *expected rank*, denoted by $er(F)$, is calculated as follows.

$$er(F) = \sum_{W \in \mathcal{W}} r(F, W) \times Pr(W) \tag{3.3}$$

where $r(F, W)$ is the rank of F in the possible world W. Then the k facilities with highest ranks are retrieved.

Given a possible world W, the rank of a facility ($r(F, W)$), is calculated based on its influence score (i.e., $s(F, W)$) as well as influence scores of other facilities, while the *expected score* computation is independent to other facilities. This implies that the *expected rank* based ranking model is much more complicate than the *expected score* based model. As shown in [ZHZZ12], we may have to enumerate all possible worlds in the worse case, which is cost-inhibitive in practise. Therefore, although novel pruning techniques are proposed to significantly improve the performance, the computational cost of the algorithm is still expensive due to the high complexity of the ranking model.

As mentioned in [CLY09], the *expected score* based ranking approach does not satisfy the *value invariance* property, which implies that the ranking results of the *expected rank* model and the *expected score* model may be different if there are

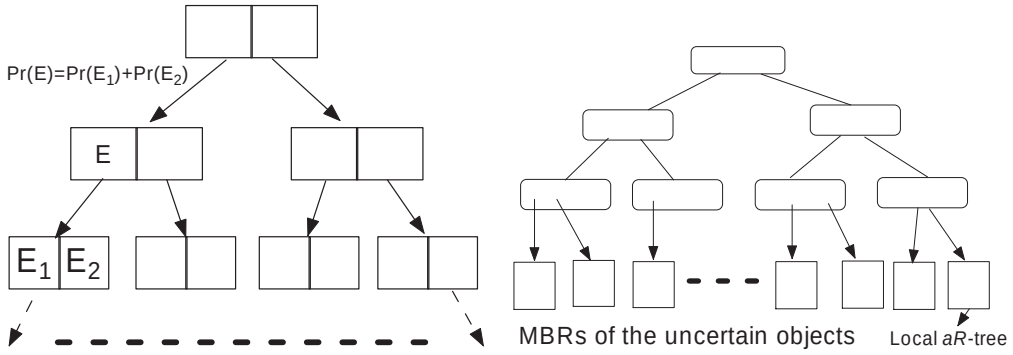
some inconsistent extreme scores in the possible worlds. For instance, a facility F has extremely high score in a few of the possible worlds such that its rank is boosted by this extreme value. Nevertheless, under our problem setting, for each possible world we have $\sum_{F \in \mathcal{F}} s(F, W) = n$ where n is the number of uncertain objects, and hence it is unlikely to have facilities with inconsistent extreme scores. This is confirmed in our empirical study which shows that two models have almost the same top k results but new algorithms proposed in this chapter are much more efficient (up to one order of magnitude faster) due to the simplicity of our new ranking model and efficiency of pruning techniques.

3.1.3 Preliminaries

Various indexing techniques have been proposed to organize uncertain objects. In this chapter, we apply R-tree and U-Quadtree based indexing techniques to facilitate the *expected influence score* computation. Note that the R-tree based indexing technique is the most widely used approach in the literature to index uncertain objects [SMP+07], and U-Quadtree is the most recent indexing technique to support range search on uncertain objects.

Indexing uncertain objects by R-tree

Given an uncertain object U, we use U_{mbr} to denote the minimal bounding rectangle (MBR) of the instances of U. Figure 3.2 illustrates the basic idea of the R-tree based indexing approach where the MBRs of the uncertain objects are indexed by R-tree [Gut84]. As to each uncertain object, an *aggregate* R-tree is employed to organize the instances where the aggregate value of each intermediate entry is the probability mass of the instances in the entry.



(a) *Local aR-tree* (b) *Global R-tree*

Figure 3.2: R-tree based Indexing

Indexing Uncertain Objects by U-Quadtree

A quadtree [FB74] is a space partitioning tree data structure in which a d-dimensional space is recursively subdivided into 2^d regions (cells). In [ZZLL12], the instances of an uncertain object U are organized by a *summary*, denoted by S_U, which consists of a set of entries $\{e\}$, where each entry records the object id ($e.oid$), the cell of the quadtree ($e.cid$) and the probability mass of instances allocated on this cell ($e.p$). The entries of the uncertain objects are organized by a $B+$ tree where the cell ids are key values, which are generated based on Hilbert curve. Figure 3.3 illustrates an example of the U-Quadtree.

Example 3.3. *In Figure 3.3(a), objects A and B have 5 instances each and all instances have the same occurrence probability (0.2). The height of the quadtree (h) is 3 and the ids of the cells are labeled. We may have $S_A = \{(1, A, 0.2), (6, A, 0.4), (8, A, 0.4)\}$ and $S_B = \{(2, B, 0.4), (6, B, 0.2), (15, B, 0.4)\}$.*

Note that the summary of an object *is not unique* as an instance x can be assigned to any cell which *contains* x. In [ZZLL12], a novel indexing construction algorithm is proposed to effectively build U-Quadtree based on the cost model. Moreover, the instances of each uncertain object are also organized by an aggregate

R-tree in [ZZLL12].

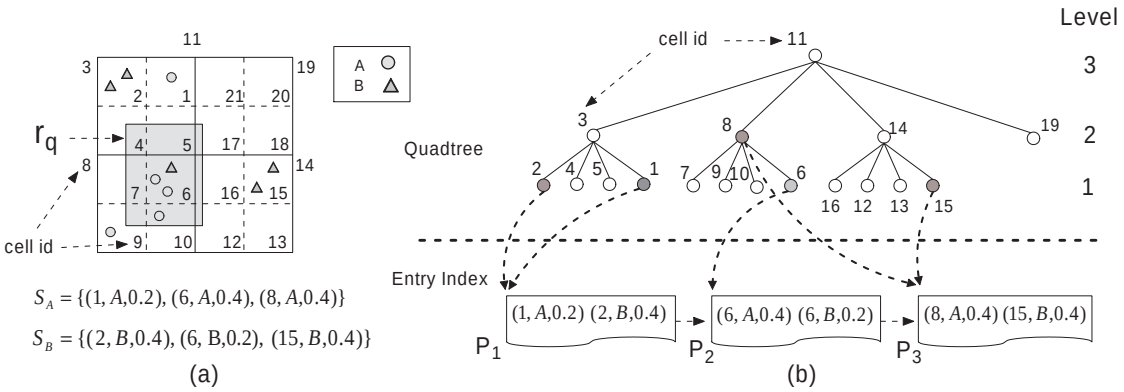


Figure 3.3: U-Quadtree

3.2 Exact Algorithms

In this section, we investigate efficient algorithms to compute the top k most influential facilities based on their *expected influence scores*. Section 3.2.1 presents a straightforward implementation of the algorithm. Assuming the uncertain objects are organized by R-tree, Section 3.2.2 improves the performance of the algorithm following the *filtering and verification* paradigm. By taking advantage of an enhanced uncertain object indexing technique, U-Quadtree, Section 3.2.3 further improves the performance of the *filtering* and the *verification* algorithms.

In this chapter, we assume facilities are organized by R-tree since it is one of the most popular index techniques in commercial spatial databases. Nevertheless, our techniques developed in this chapter can be easily extended to other hierarchical spatial indexing techniques.

3.2.1 Naive Algorithm

Algorithm 1 illustrates a naive implementation of the algorithm to compute the nearest neighbor probability of each instance regarding all facilities following Equation 3.2. For each instance of an uncertain object, a nearest neighbor query [PM97] is issued to find its nearest facility F and the *expected score* of F is increased by the occurrence probability of the instance.

Algorithm 1: ***Naive Algorithm***(S_U, S_F, k)

Input : k, Uncertain object set S_U, Facility set S_F

Output: Top k most influential facilities

```
1 for each U ∈ S_U do
2     for each instance u ∈ U do
3         for find nearest facility F ∈ S_F do
4             I(F) := I(F) + p_u;
5 return top k facilities with highest expected score;
```

Although we do not need to explore all possible worlds following the *expected score semantics*, the performance of the algorithm is not scalable to the number of instances and facilities because the instances of all objects are accessed in Algorithm 1 and the *expected scores* are calculated for all facilities, which leads to high CPU and I/O costs.

3.2.2 R-tree based Algorithm

To address the scalability issue in the above naive algorithm, in this subsection, we propose the R-tree based algorithm following the *filtering and verification* paradigm. More specifically, based on the MBRs of the uncertain objects, which

are organized by R-tree, we can come up with the lower and upper bounds of the *expected influence scores* of the facilities in the *filtering* phase. Then some facilities can be pruned based on the widely used top k filtering conditions; that is, a facility F will be eliminated from candidate set if there are a set $\mathcal{Q}$ of k other facilities such that $I^+(F) < I^-(F')$ for any facility F' in $\mathcal{Q}$, where $I^+(F)$ ($I^-(F)$) denotes the upper (lower) bound of *expected score* for the facility F. In the *refinement* phase, we only need to explore the instances of the uncertain objects which may contribute to the *expected scores* of the facilities in the candidate set.

In this chapter, we assume the MBRs of the uncertain objects and facilities are organized by an aggregate R-tree $R_{\mathcal{O}}$ and a R-tree $R_{\mathcal{F}}$ respectively. We first introduce some notations used in this chapter.

Definition 3.1. *Nearest Neighbor Distance (NND). Given a set of facilities $\mathcal{F}$, the distance between a point x and its nearest neighbor F is the nearest neighbor distance of x regarding $\mathcal{F}$, denoted by $nnd(x, \mathcal{F})$. In [XZKD05], effective method is proposed to compute the minimal and maximal nearest neighbor distances between two rectangles. In this chapter, we use $nnd_{min}(R_1, R_2)$ to denote the minimal nearest neighbor distance between two rectangles R_1 and R_2; that is, for any point x in R_1, its nearest neighbor distance regarding a set of facilities $\mathcal{F}$ contained by R_2 is not smaller than $nnd_{min}(R_1, R_2)$, i.e., $nnd(x, \mathcal{F}) \geq nnd_{min}(R_1, R_2)$. With the same rationale, we have $nnd_{max}(R_1, R_2)$ where $nnd(x, \mathcal{F}) \leq nnd_{max}(R_1, R_2)$.*

To enable computing the *expected scores* of the facilities in a level by level fashion, we introduce the concept of object tuple and facility tuple so that the *expected score* of a group of facilities or uncertain objects can be updated or pruned at the same time.

Definition 3.2. *Object Tuple (T). An object tuple T is employed to maintain the information used for the NND computation of a set of uncertain objects in an entry*

e. Particularly, T.e is the object R-tree entry (intermediate entry or data entry) associated with T, and T.$\mathcal{F}$ is a set of facility R-tree entries (intermediate entry or data entry) which may contribute to the NND computation of the objects in T.e.

Example 3.4. *In Figure 3.4, there are four uncertain objects U_1, U_2, U_3 and U_4, and their MBRs are kept in data entries $\{e_1, e_2, e_3, e_4\}$. Suppose the object tuple T refers to the entry E_1, then we have $T.e = E_1$ and $T.\mathcal{F} = \{F_1, F_2, F_3\}$ where F_1, F_2, F_3 are facilities entries which may contribute to the NND computation of the object associated with T.*

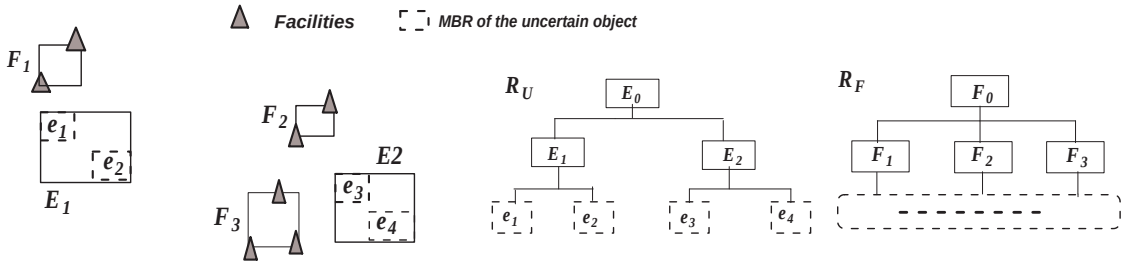


Figure 3.4: Motivation Example

Whenever there is no ambiguity, we use F to denote an entry in the facility R-tree. We use $I^-(F)$ and $I^+(F)$ to denote the lower and upper bounds of the *expected score* of F. Note that F may represent an intermediate entry which contains a set of facilities.

R-tree based Filtering

Motivation

The basic idea of the filtering algorithm is to conduct the *NND* computation on the high level entries of $R_{\mathcal{O}}$ and $R_{\mathcal{F}}$ such that we do not need to compute the *NND* regarding each individual object and facility, and hence improve the performance of the algorithm in terms of CPU and I/O costs.

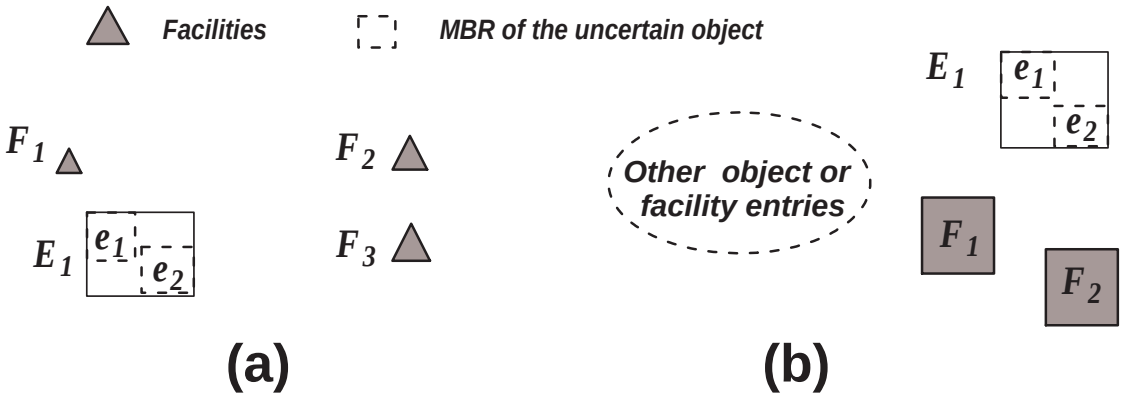


Figure 3.5: Motivation Example

In Figure 3.4, let T refer to the entry E_1 and $T.\mathcal{F} = \{F_1, F_2, F_3\}$. We can remove F_2 and F_3 from $T.\mathcal{F}$ since the maximal *NND* between E_1 and F_1 ($nnd\_max(E_1, F_1)$) is smaller than the minimal *NND* from E_1 to F_2 and F_3 ($nnd\_min(E_1, F_1)$ and $nnd\_min(\ E_1,\ F_3)$) , which implies that none of the facilities in F_2 and F_3 can contribute to the *NND* computation of the objects in E_1 and hence $T.\mathcal{F} = \{F_1\}$. Similarly, we have $T.\mathcal{F} = \{F_2, F_3\}$ when $T.e$ refers to E_2. Moreover, let λ be the k-th largest lower bounds of the *expected scores* for the facility entries seen so far, we do not need to further explore the entries since none of the facilities in the entry can be top k influential facilities. In this chapter, we say a facility entry F is *disabled* if $I^+(F) < \lambda$.

Besides the facility entries, we can also *prune* object entries in this chapter. In Figure 3.5(a), we have $T.e = E_1$ and $T.\mathcal{F} = \{F_1, F_2, F_3\}$. Suppose F_1 is a data entry, i.e., F_1 corresponds to a single facility, and the maximal *NND* between F_1 and E_1 is smaller than the minimal *NND* from E_1 to F_2 and F_3, then we can increase $I^-(F_1)$ by $agg(E_1)$ where $agg(E_1)$ is the number of uncertain objects in E_1. Clearly, we do not need to further explore the uncertain objects in E_1 and E_1 is marked as *disabled*. On the other hand, as shown in Figure 3.5(b), suppose all facility entries in $T.\mathcal{F}$ are *disabled* (shown as grey rectangles in the example), we

can also prune E_1 since objects in E_1 only contribute to the *expected scores* of the non-promising facilities.

Algorithm Algorithm 2 illustrates the details of the R-tree based *filtering* algorithm on $R_{\mathcal{O}}$ and $R_{\mathcal{F}}$, which follows the *synchronized* R-tree traversal paradigm used in spatial joins. A FIFO queue (Q) is employed to keep object tuples, and the first object tuple is initialized by the roots of $R_{\mathcal{O}}$ and $R_{\mathcal{F}}$ (Lines 2-5). For each object tuple T popped from Q, Line 10 puts the object tuple T to $\mathcal{S}$ which keeps the objects that need to be further explored in the *refinement* phase **if** all entries in $T.e$ and $T.\mathcal{F}$ are *data* entries. Otherwise, Line 13 generates an object tuple t for each child entry of $T.e$ [1]. Then for each facility entry in $T.\mathcal{F}$, we put all of its child entries $\{e_f\}$ to $t.\mathcal{F}$ if it is not marked as *disabled*. Meanwhile, $I^-(e_f)$ and $I^+(e_f)$ are set by its parent entry in Line 18. Line 19 calculates the maximal *NND* from $T.e$ to facility entries in $t.\mathcal{F}$, denoted by d_{max}. For each facility entry e_f in $t.\mathcal{F}$, we exclude e_f from $t.\mathcal{F}$ if $nnd\_min(t.e, e_f) > d_{max}$ and decrease $I^+(e_f)$ by $agg(t.e)$ (Line 20-23). Recall that $agg(t.e)$ is the number of uncertain objects in $t.e$. The facility entry e_f will be pruned in Line 25 if its upper bound of the *expected score* is smaller than λ. In Line 26-28, we increase $I^-(e_f)$ by $agg(t.e)$ and do not further explore uncertain objects in $t.e$ (i.e., prune t) if e_f is the only **data** entry in $e.\mathcal{F}$. In Line 29-30, we prune the object tuple t if all of the facilities in $t.\mathcal{F}$ are non-promising facilities. Line 32 pushes the object tuple t to Q if it is not *disabled*. Finally, Algorithm 2 terminates when Q is empty, and the facilities surviving the *filtering* phase ($\mathcal{C}$) will be returned as well as the object tuples in $\mathcal{S}$.

[1] In the case that $T.e$ is a data entry, we simply set $t.e = T.e$ at Line 13. The same strategy goes for facilities in Line 16.

Algorithm 2: ***R-tree based Filtering*** $(R_{\mathcal{O}}, R_{\mathcal{F}}, k)$

Input : $R_{\mathcal{O}}$: the aggregate R-tree for uncertain objects,
$R_{\mathcal{F}}$: the R-tree for facilities, k
Output: $\mathcal{C}$: a set of candidate facilities,
$\mathcal{S}$: objects need to be further explored

```
1  C := ∅; S := ∅; Q := ∅; λ = 0;
2  generate a new tuple T;
3  T.e ← the root of R_O;
4  e_f ← the root of R_F;
5  T.F ← e_f; I⁻(e_f) := 0 ; I⁺(e_f) := # objects in R_O;
6  push T into FIFO queue Q;
7  while Q is not empty do
8      T ← pop the head of Q;
9      if T.e is data entry and all facility entries in T.F are data entries then
10         S := S ∪ T;
11     else
12         for each child entry e' of T.e do
13             generate a new object tuple t for e' where t.e := e';
14             for each facility entry F in T.F do
15                 if F is not disabled then
16                     for each child entry e_f of F do
17                         t.F := t.F ∪ e_f;
18                         I⁻(e_f) := I⁻(F); I⁺(e_f) := I⁺(F);
19             d_max := min{nnd_max(t.e, e_f) } for all e_f in t.F;
20             for each facility entry e_f in t.F do
21                 if nnd_min(t.e, e_f) > d_max then
22                     t.F := t.F − e_f;
23                     I⁺(e_f) := I⁺(e_f) − agg(t.e);
24                 if I⁺(e_f) < λ then
25                     Disable F;
26             if t.F contains exactly one data entry e_f then
27                 I⁻(e_f) := I⁻(e_f) + agg(t.e);
28                 Update λ; Disable t;
29             if all facilities in t.F are disabled then
30                 Disable t;
31             if t is not disabled then
32                 Push t to the tail of Q;
33 C ← facilities with I⁺(F) ≥ λ;
34 return S, C;
```

R-tree based Refinement

After the *filtering* phase, we need to explore the instances of the uncertain objects in $\mathcal{S}$ such that we can come up with the top k influential facilities in the *refinement* phase. Algorithm 3 illustrates the framework of the *refinement* procedure. For each object tuple T in $\mathcal{S}$, we apply the function **Refinement** to refine the *expected scores* of the facilities in the candidate set. Note that we do not need to process T if all facilities in $T.\mathcal{F}$ are marked as *disabled*. Finally we have $I(F) = I^{-}(F)$, and the k facilities with the highest *expected scores* are retrieved.

Algorithm 3: ***R-tree based Refinement***$(\mathcal{C}, \mathcal{S}, k)$

Input : $\mathcal{C}$: the candidate facilities,
$\mathcal{S}$: the objects tuples, k

Output: $\mathcal{I}$: the top k influential facilities

```
1 I := ∅;;
2 for each object tuple T in S do
3     if all facilities in T.F are disabled then
4         Goto Line 2;
5     U ← the uncertain object associated with T.e;
6     Refinement( root of R_U, T.F);
7 I ← k facilities with the highest I⁻(F) values;
8 return I;
```

In the following, we first discuss the access orders at Line 2 of Algorithm 3 , then present the function **Refinement** at Line 6.

Access Order. Intuitively, we should put high priority to the objects which contribute to the facilities with large upper bounds of the *expected scores* since they are more likely to be the top k influential facilities, and hence leads to a

tighter *expected score* threshold λ, i.e., better pruning power. In this chapter, an object tuple T is sorted by the largest upper bounds of *expected scores* for facilities in $T.\mathcal{F}$. Our empirical study shows this strategy outperforms other alternatives such as the random order and ordering by the size of $T.\mathcal{F}$, i.e., the number of facilities which may influence the uncertain objects associated with $T.e$.

Refinement Algorithm. Algorithm 4 is used to update the *expected scores* of the facilities by exploring the instances kept in an aggregate R-tree entry e. It is similar to the R-tree based filtering algorithm (Algorithm 2) except that: (i) In Algorithm 2, both object entry and facility entry are drilled down in a level by level fashion, while in Algorithm 4 only the object entries are expanded since we already reach the bottom of the facility R-tree in the *filtering* phase. (ii) At Line 7, 10, 18 and 22 of Algorithm 4, $p(e)$ represents the probability mass of the instances in the aggregate R-tree entry e.

3.2.3 U-Quadtree based Algorithm

Observe that the performance of R-tree technique is poor when the sizes of the MBRs of the uncertain objects are not very small because it is not effective to capture the instance distribution of an uncertain object by a single MBR. In [ZZLL12], Zhang *et al.* propose a novel indexing structure based on the quadtree such that a good tradeoff can be achieved between the filtering cost and refinement cost.

Suppose the uncertain objects are organized by U-Quadtree and the instances of each uncertain object are kept in an aggregate R-tree, in this subsection, we present efficient U-Quadtree based algorithm to identify the top k influential facilities.

Algorithm 4: ***Refinement***$(e, \mathcal{F})$

Input : e : the R-tree entry,
$\mathcal{F}$: a set of facilities
Output: Updated $\mathcal{F}$

```
1  Q := ∅; T.e := e ; T.F := F;
2  push T to Q;
3  while Q is not empty do
4      T ← pop the head of Q;
5      if T.e is a data entry then
6          Find the nearest facility F in T.F;
7          I⁻(F) := I⁻(F) + p(T.e);
8          Update λ;
9          for other facility F' in T.F do
10             I⁺(F') := I⁺(F') − p(T.e);
11             Disable F' if I⁺(F') < λ;
12     else
13         for each child entry e' of T.e do
14             t.e ← e';
15             t.F := T.F;
16             d_max := min{nnd_max(T.e, F) | F ∈ T.F};
17             for facility F in t.F with nnd_min(T.e, F) > d_max do
18                 I⁺(F) := I⁺(F) − p(t.e);
19                 remove F from t.F;
20                 Disable F if I⁺(F) < λ;
21             if There is only one facility F in t.F then
22                 I⁻(F) := I⁻(F) + p(t.e);
23                 Update λ;
24             else
25                 Push t on the tail of Q;
26 return F , λ;
```

U-Quadtree based Filtering

As the U-Quadtree is also a hierarchical spatial tree structure, Algorithm 2 can be modified to support the *filtering* procedure. Recall that the MBR of an uncertain object U is kept for R-tree index, while an uncertain object U is described by a summary S_U in U-Quadtree which consists of a set of entries where each entry is represented by its corresponding cell in the quadtree and the probability mass of the instances assigned to the cell. Therefore, in Line 10 of Algorithm 2, an object tuple is kept in $\mathcal{S}$ if it is a cell at the lowest level and all facilities in $T.\mathcal{F}$ are data entries. Moreover, for an object tuple T, $T.e$ corresponds to a cell c which maintains the entries of the uncertain object summaries assigned to c, and $T.\mathcal{F}$ records the facility entries which may influence the instances of the uncertain objects associated with c. When $I^-(e_f)$ ($I^+(e_f)$) is increased (decreased), instead of $agg(T.e)$ the probabilities sum of all entries on c (i.e., $T.e$) will be used.

U-Quadtree based Refinement

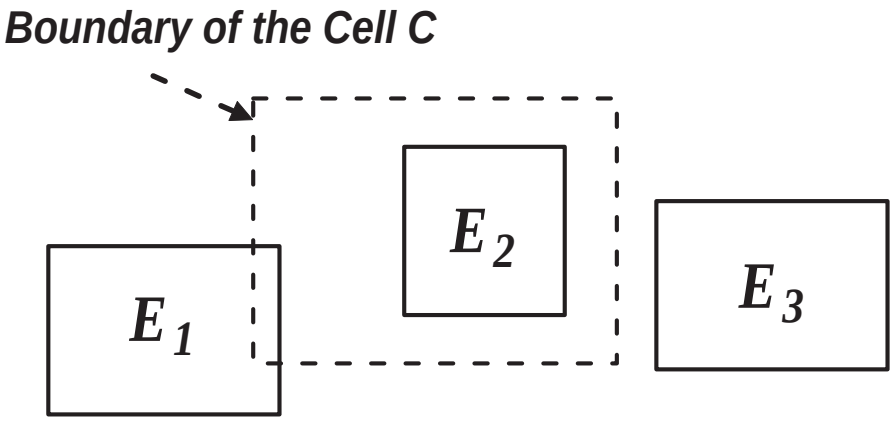


Figure 3.6: Containment Example

We first introduce the *containment* relationship between a cell c and an entry E in our algorithm description. The cell c *fully contains* the entry E if all points

in the MBR of E are contained by the boundary of c. If c does not *fully contain* E but some points in the MBR of E are contained by the boundary of c, then c *partially contains* E. Otherwise, there is no containment relationship between c and E. For instance, in Figure 3.6, E_2 is *fully contained* by c and E_1 is *partially contained* by c.

Algorithm 5: ***U*-Quadtree based Refinement** ($\mathcal{C}$, $\mathcal{S}$, k , λ)

Input : $\mathcal{C}$: the candidate facilities,
$\mathcal{S}$: the objects need to further explore,
k: top k

Output: $\mathcal{I}$: the top k influential facilities

```
1  I := ∅;
2  for each uncertain object U in S do
3      for each cell c associated with U do
4          Q := ∅; Q ← root of R_U;
5          while Q is not empty do
6              e ← pop the head of Q;
7              if MBR of e is fully contained by the cell c then
8                  Refinement( e , F);
9              else if MBR of e is partially contained by the cell c then
10                 for each entry e' of e do
11                     Push e' to the tail of Q;
12 I ← k facilities with the highest I⁻(F);
13 return I;
```

Algorithm 5 illustrates the details of the U-Quadtree based refinement algo-

rithm. For each uncertain object U survived in the *filtering* phase, we issue a set of window queries to update the *expected scores* of the facilities in $\mathcal{F}$. More specifically, a FIFO queue Q is employed to keep entries in the aggregate R-tree of the uncertain object U (R_U), which is initialized by the root of R_U. For an entry e in the R_U, we will invoke the function **Refinement if** it is *fully contained* by c. Otherwise, the child entries of e are pushed to Q **if** e is *partially contained* by c. Note that a *data* entry will either be *fully contained* by c or have no containment relationship with c. Finally, we have $I(F) = I^{-}(F)$, and the k facilities with the highest *expected scores* are returned.

3.3 Randomized Algorithms

In some scenarios, each object may contain a large number of instances. For example, an uncertain object may appear at many possible locations. Moreover, to support the query processing on uncertain objects which are described by continuous probabilistic density functions, a massive number of instances will be sampled to discretize the *continuous* uncertain object (e.g., [TXC07]). As expected, it is reported in our empirical study that the performance of exact algorithms degrades when the number of instances grows due to the expensive I/O costs incurred. Therefore, in this Section we propose randomized algorithms to significantly improve the performance with accuracy guarantee. Specifically, we briefly introduce the motivation of the randomized algorithm as well as its straightforward implementation in Section 3.3.1. Section 3.3.2 shows that the accuracy of the algorithms are guaranteed with sufficient number of sampled instances for each uncertain object. In Section 3.3.3, we show that the exact algorithms developed in Section 3.2 can be immediately applied against the sampled instances of the uncertain objects to

improve the performance.

3.3.1 Motivation

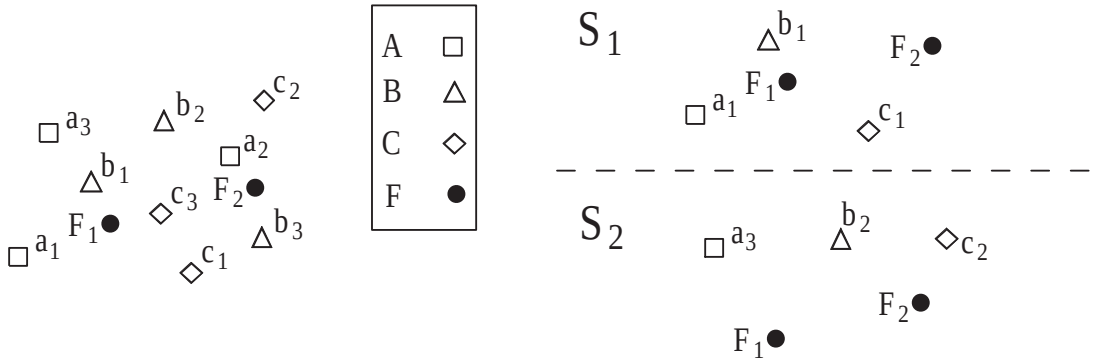
The basic idea of the randomized algorithm is to sample all possible worlds $\mathcal{W}$ by a small number of s possible worlds S_1, S_2, . . ., S_s. Given n uncertain objects, each sampled possible world S_i consists of n sampled instances where an instance is randomly sampled from each uncertain object. In each sample set S_i (i.e., the i-th sampled possible world), the sampled instance of an uncertain object is chosen according to their appearance probabilities; that is, an instance $u \in U$ is chosen with probability p_u in S_i. For each sample set S_i, we can calculate the influence score of each facility with traditional approach since there is only one instance for each uncertain object regarding S_i. Then we can come up with the estimation of the influence score for each facilities by the average scores of the facilities on the samples sets S_1, S_2, ..., S_s; that is, given sample sets $\mathcal{S}$, we can use the following formula to estimate the *expected influence score* of a facility F where $\tilde{I}_i(F)$ is the number of reverse nearest neighbors (i.e., influence score) of F in the sample set S_i.

$$\tilde{I}(F) = \sum_{i=1}^{s} \tilde{I}_i(F)/s \tag{3.4}$$

Following is an example of the randomized method.

Example 3.5. *As shown in Figure 3.7, suppose there are three objects and each object has three instances with the same appearance probability. There are two sample sets S_1 and S_2 where $S_1 = \{a_1, b_1, c_1\}$ and $S_2 = \{a_3, b_2, c_2\}$. According to Equation 3.4, we have $\tilde{I}_1(F_1) = 2$, $\tilde{I}_1(F_2) = 1$, $\tilde{I}_2(F_1) = 1$, and $\tilde{I}_2(F_2) = 2$.*

Algorithm 6 illustrates a straightforward implementation of the randomized algorithms for *expected score* estimation according to Equation 3.4. For each sample



(a) Uncertain Objects (b) Two Sample sets S 1 and S 2

Figure 3.7: Example of *Sampling* Approach

set $S_i \in \mathcal{S}$ where $\mathcal{S}$ represents s sample sets, we can calculate the influence score for each facility. Line 4 increases the estimated influence score of a facility F, denoted by $\tilde{I}(F)$, by 1 against each of its reverse nearest neighbors.

In Section 3.3.2 we conduct detailed analysis on the accuracy guarantee of Algorithm 6. Then we further improve the performance of the randomized algorithm in Section 3.3.3.

Algorithm 6: *Naive Randomized Algorithm*($\mathcal{S}$, S_F, k)

Input : Sample sets $\mathcal{S} = \{S_1, \ldots, S_s\}$, Facility set S_F, k

Output: Top k most influential facilities

```
1 for each S_i ∈ S do
2     for each sampled instance u ∈ S_i do
3         for find its nearest facility F ∈ S_F (i.e.,NN(u) = F) do
4             Ĩ(F) := Ĩ(F) + 1;
5 for each facility F ∈ S_F do
6     Ĩ(F) := Ĩ(F)/s;
7 return top k facilities with highest estimated expected influence score;
```

3.3.2 Accuracy Evaluation

In this subsection, we show that with sufficient number of sample sets, our randomized algorithm can achieve a good accuracy. For an uncertain object U_j, let $u_{i,j}$ denote the sampled instance of U_j in S_i where $1 \leq i \leq s$ and $1 \leq j \leq n$. Then for a facility F_l, the event that the nearest facility of $u_{i,j}$ is F_l in S_i, i.e., $NN(u_{i,j}) = F_l$, can be described by the following random variable.

$$Z_{i,j,l} = \begin{cases} 1 \;\; if \;\; NN(u_{i,j}) = F_l \;\; in \;\; S_i \\ 0 \;\; otherwise \end{cases} \tag{3.5}$$

Let the random variable $Y_{i,l}$ record the influence score of the facility F_l in S_i where

$$Y_{i,l} = \sum_{j=1}^{n} Z_{i,j,l} \tag{3.6}$$

Similarly, we define the random variable X_l where

$$X_l = \sum_{i=1}^{m} Y_{i,l}/m \tag{3.7}$$

According to the description of Algorithm 6 and Equation 3.4, we have $\tilde{I}(F_l) = X_l$ and $\tilde{I}_i(F_l) = Y_{i,l}$.

Since $Pr(Z_{i,j,l} = 1) = Pr(NN(u_{i,j} = F_l))$ where $u_{i,j}$ is the sample instance of the uncertain object U_j in the sample set S_i, we have

$$E(Z_{i,j,l}) = \sum_{u \in U_j \;\wedge\; NN(u) = F_l} p_u \tag{3.8}$$

Consequently, according to the definition of $I(F)$ in Equation 3.2, we have

$$E(Y_{i,l}) = E(\sum_{j=1}^{n} Z_{i,j,l}) = I(F_l) \tag{3.9}$$

and

$$E(X_l) = E(\sum_{i=1}^{s} Y_{i,l}/s) = I(F_l) \tag{3.10}$$

Because the random variables $\{X_{i,j,l}\}$ are independent to each others, we have the following theorem based on the Hoeffding bound [Hoe63].

Theorem 3.1. *Given an* $\epsilon(0 < \epsilon < 1)$*, a* $\delta(0 < \delta < 1)$*, and* n *objects, if* $s = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$*, then*

$$Pr(|X_l - I(F_l)| \leq \epsilon n) \geq 1 - \delta$$

Theorem 3.1 indicates that when the number of sampling sets (s) is sufficiently large, our estimation of $I(F)$ (i.e., $\tilde{I}(F)$) in Algorithm 6 is rather accurate.

As we aim to retrieve the top k most influential facilities, it is more interesting to evaluate the precision and recall of the top k answer returned. Since we only return k facilities, the precision and recall are always the same in Algorithm 6. In the following, we show that when the number of sample sets s is sufficiently large, the precision and recall can be guaranteed with probability at least $1 - \delta$.

We first introduce the following Lemma.

Lemma 3.1. *For two facilities* F_i *and* F_j*, suppose that* $I(F_i) > I(F_j)$*. Then,* $Pr(X_i \leq X_j) \leq exp(-s(I(F_i) - I(F_j))^2/(2 * n^2))$.

Proof. Let $Z = X_j - X_i$. We have $Pr(X_j \leq X_i) = Pr(Z - E(Z) > I(F_i) - I(F_j))$. By Hoeffding inequality (Theorem 2 in [Hoe63]), the lemma is immediate. □

Suppose we have $I(F_i) > I(F_j)$ (i.e., $E(X_i) > E(X_j)$), Lemma 3.1 indicates the likelihood that the order of X_i and X_j is reversed (i.e., $X_i < X_j$) in Algorithm 6 decreases exponentially against the sample set size s and the difference between $I(F_i)$ and $I(F_j)$. Based on the above observation, we have the following theorem which indicates that Algorithm 6 can achieve a good precision and recall when the number of the sample sets s is sufficiently large.

Theorem 3.2. *For a given precision or recall α, suppose the objects are sorted on their expected scores with non-increasing order. Let λ equal to $I(F_\beta) - I(F_{k+1})$ where $\beta = \lceil \alpha k \rceil$ and assume $\lambda \geq \epsilon n$. Then when the number of sample sets $s = O(\frac{2}{\epsilon^2} log \frac{k \times n}{\delta})$, we have $Pr(k^* < \alpha \times k) \leq \delta$ where k^* is the number of* true *top k most influential facilities returned by Algorithm 6.*

Proof. As shown in Figure 3.8, if none of the facilities in $F_{k+1}, \ldots, F_n$ reverse order with any facilities in $F_1, \ldots, F_\beta$, we have $k^* > \alpha \times k$. Since the probability of $X_{k+1} > X_\beta$ is bounded by $exp(-s\lambda^2/n^2)$ based on Lemma 3.1 and for all $l < \beta$ and $j > k+1$, $Pr(X_j > X_l) \leq Pr(X_{k+1} > X_\beta)$, we have $Pr(k^* < \alpha \times k) < k \times n \times exp(-s \times \lambda^2/(2n^2)) \leq \delta$. □

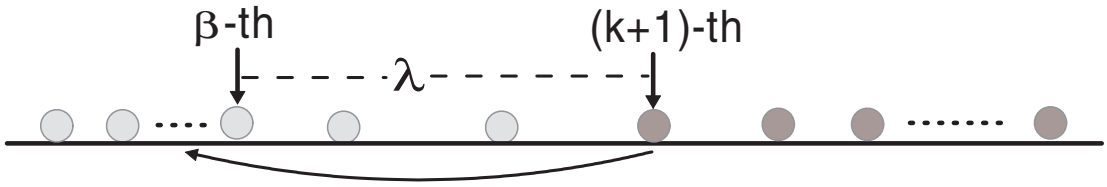


Figure 3.8: Proof of Theorem 3.2

3.3.3 Enhanced Randomized Algorithms

Because of the small number of sample sets, Algorithm 6 can improve the performance when the number of instances in each object is massive. In this subsection, we discuss how to further improve the performance of randomized algorithms by applying indexing techniques on the sampled instances.

Intuitively, for each sample set (round) S_i we can organize the sampled instances by an R-tree. Then the all nearest neighbor computation techniques (e.g., [CP07]) can be immediately employed to speed up the computation of influence scores in Algorithm 6. However, all sampled instances will be explored which may lead to

high I/O costs. To address this issue, we organize the sampled instances in another way.

Observe that the calculation of $\tilde{I}(F_l)$ for a facility F_l in Algorithm 6 can be rewritten as follows due to the independence of the random variables $\{Z_{i,j,l}\}$.

$$\begin{aligned} \tilde{I}(F_l) &= \sum_{i=1}^{s}\sum_{j=1}^{n} Z_{i,j,l}/s \\ &= \sum_{j=1}^{n}\sum_{i=1}^{s} Z_{i,j,l}/s \end{aligned} \tag{3.11}$$

Equation 3.11 implies that we can compute the influence score for each individual facility, and hence the sampled instances from the **same object** can be organized in the same spatial indexing structure, instead of grouping sampled instances in the **same sample set**. According to Equation 3.2 and Equation 3.5 as well as the fact that an instance u of the uncertain object U is chosen with probability p_u, we can apply the exact algorithms developed in Section 3.2 by regarding s sampled instances of an object U as its instances; that is, we have $U = \{u_1, \ldots, u_s\}$ and $p_{u_i} = \frac{1}{s}$ where u_i is the sampled instance of U in S_i. Our empirical study shows that this can significantly improve the performance of the randomized algorithm.

3.4 Experiment

In this section, we present results of a comprehensive performance study to evaluate the efficiency and scalability of the proposed techniques in the chapter. Following algorithms are evaluated.

- **Naive** The naive implementation proposed in Section 3.2.1.
- **RTKIS** The technique based on R-tree proposed in Section 3.2.2.

- **UQuadTKIS** The technique based on U-Quadtree proposed in Section 3.2.3.
- **UTKIS** The technique presented in [ZHZZ12].
- **Naive*** The naive randomized algorithm proposed in Section 3.3.1.
- **RTKIS* and UQuadTKIS*** The enhanced randomized algorithms proposed in Section 3.3.3, which are based on **RTKIS** and **UQuadTKIS** algorithms respectively.

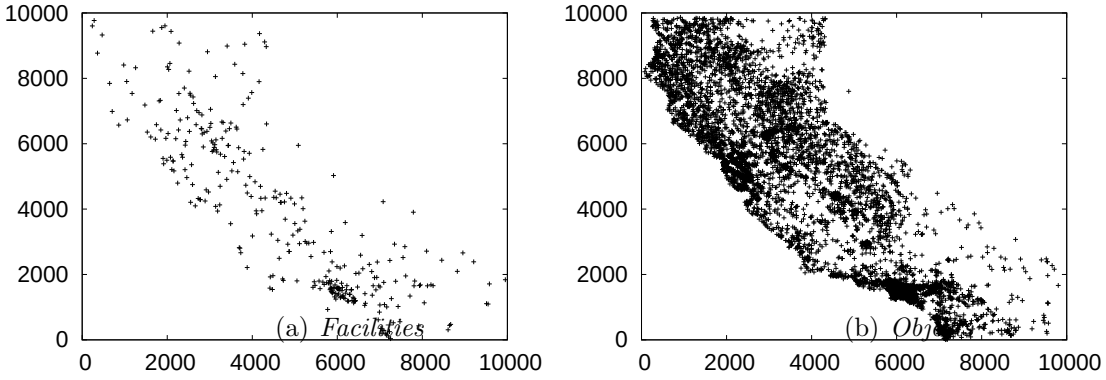


Figure 3.9: Data distribution

Datasets. Three real spatial datasets, namely CA, USA and RT, are used to evaluate our techniques. CA and USA contain $62K$ and $200K$ 2-dimensional points representing locations in the Los Angeles and the United States respectively which are available at [Bur]. The two datasets are separated into several groups of data respectively, and we choose one group of data with 996 points as the facilities and the other group of data with $21,050$ points to represent the centers of uncertain objects from CA as default dateset, whose distributions are showed in Figure 3.9. RT is obtained from the R-tree-Portal [The03] with cultural landmarks and populated places in North America. The number of uncertain objects in USA and RT are $20,287$ and $24,493$ respectively. By default, around 1000 facilities are chosen

from corresponding datasets. In our experiment, all dimensions are normalized to domain [0, 10000], and the uncertain region of the uncertain object is a circle with expected radius r_u varying from 20 to 300 with default value 60. There are m instances for each uncertain object and the expected m varies from 100 to 500 with default value 200. Therefore, the total number of instances in default dataset is $4,210,000$. The instances of an uncertain object follow popular distributions Normal(N) and Uniform(U) where Normal(N) distribution serves as default instance distribution. To evaluate randomized algorithms, we generate a large number of instances m_r for each uncertain object and the expected m_r varies from 500 to 4K with default value 2K, and the number of sample s varies from 20 to 400 with default value 200.

All algorithms proposed in this chapter are implemented in standard C++ with STL library support and compiled with GNU GCC. Experiments are run on a PC with Intel Xeon 2.40GHz dual CPU and 4G memory running Debian Linux. The disk page size is fixed to 4096 bytes and the capacity of the entry page (f) is set to 512. In the chapter, we evaluate the I/O performance of the algorithms by measuring the number of uncertain objects explored, i.e., uncertain objects whose aggregate R-tree are loaded in main memory. Query response time is recorded to evaluate the efficiency of the algorithms, which contains the CPU time and the I/O latency.

Table 3.2 lists parameters which may have an impact on our performance study (default values are listed as **Bold**). In our experiments, all parameters use default values unless otherwise specified.

Notation	Definition
height of U-Quadtree h	5, 6, 7, 8, **9**, 10, 11
radius of uncertain object region r_u	20, **60**, 100, 200, 300
number of uncertain objects n	10K, **20K**, 30K, 40K, 50K
number of instances m	100, **200**, 300, 400, 500
number of facilities f	200, 400, 600, 800, **1000**
top k	10, **20**, 30, 40, 50
number of instances-large data m_r	500, 1K, **2K**, 3K, 4K
number of sample-large data s	20, 50, 100, **200**, 400
instance location	**normal**, uniform

Table 3.2: Parameter settings

3.4.1 Performance Tuning

The performance of UQuadTKIS is effected by the height (h) of the U-Quadtree. As expected, Figure 3.10 shows that number of objects visited in the *refinement* phase drops when h increases due to the larger size of uncertain object summaries. Nevertheless, UQuadTKIS becomes less efficient for larger h when $h > 9$, which implies that the algorithm cannot pay-off the larger index size when $h > 9$. In the following experiments, h is set to 9.

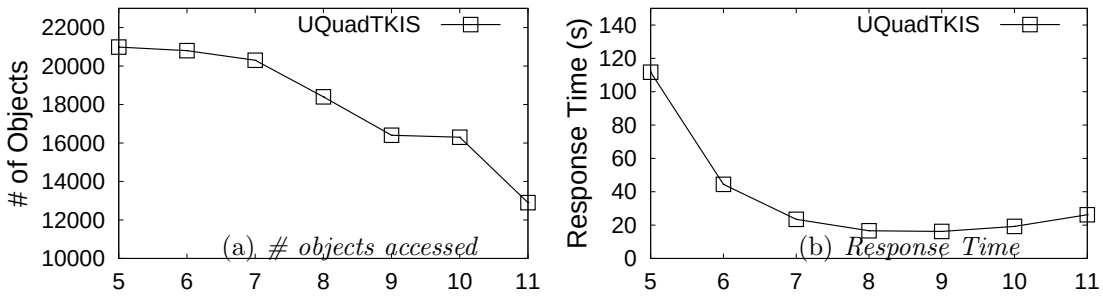


Figure 3.10: Diff. U-Quadtree Height

Figure 3.11 reports the effectiveness of different access order strategies in R-tree based *refinement* algorithm where the sizes of the radius grow from 20 to 100. Particularly, "by fac" denotes the facilities *expected score* based access order

strategy used in Algorithm 2, and "by obj" stands for the object based strategy, i.e., accessing in decreasing order of the number of facilities associated with each object. It is shown that our facility based strategy always outperforms the object based strategy.

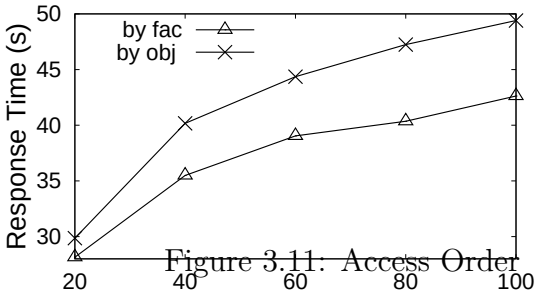


Figure 3.11: Access Order

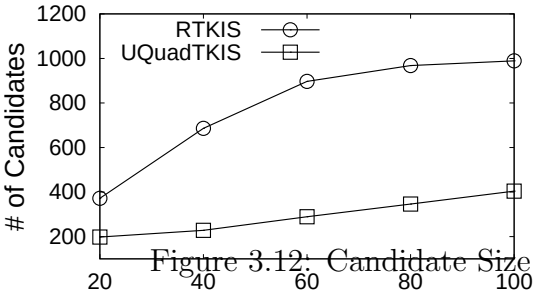


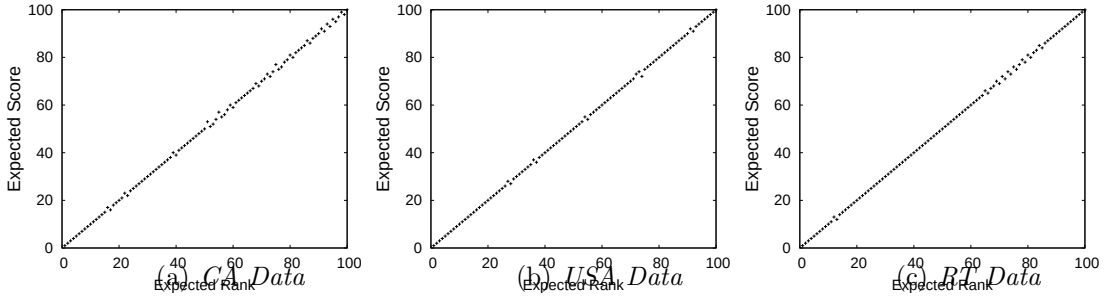
Figure 3.12: Candidate Size

We evaluate the *filtering* effectiveness of RTKIS and UQuadTKIS in Figure 3.12 by measuring the number of candidates (facilities) after *filtering* phase. The performance of both algorithms degrade against the growth of r_u. UQuadTKIS significantly outperforms RTKIS since more resources are allocated to the U-Quadtree to capture the distribution of the instances of the uncertain objects.

3.4.2 Exact Algorithm Performance Evaluation

We will evaluate the performance of four exact algorithms (Naive, RTKIS, UQuadTKIS, and UTKIS) in this sub-section.

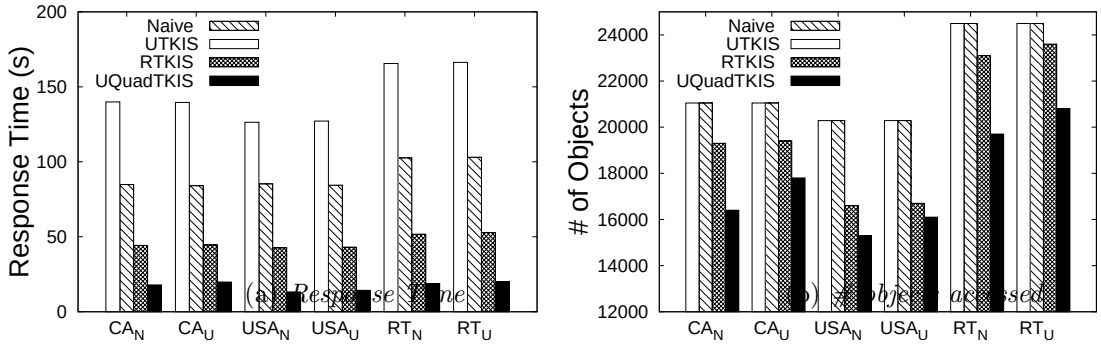
Comparing Different Ranking models. Figure 3.13 evaluates the similarity of different ranking models (*expected rank* and *expected score*) on three datasets CA, USA and RT respectively, which shows the scatter-plot of the ranks of 100 facilities. Particularly, each point in the scatter-plot represents a facility where x-axis and y-axis record the rank of objects based on *expected rank* model and *expected score* model respectively. It is shown that all points line up along the diagonal, and



(a) *CA Data* (b) *USA Data* (c) *RT Data*

Figure 3.13: Result comparison

the maximal difference of the ranks for a facility is only 2 in Figure 3.13 , which indicates that the results of two models are almost the same.



(a) *Response Time* (b) *# objects accessed*

Figure 3.14: Impact of Data Distributions

Impact of Data Distribution. Figure 3.14 reports the response time and the number of uncertain objects accessed of the algorithms against different data distributions where CA_N represents the dataset in which the centers of uncertain objects are from CA and the instances of each uncertain object follow the Normal(N) distribution. It is reported that UQuadTKIS significantly outperforms other algorithms under all data distributions, and UTKIS ranks the last due to the high complexity of the *expected rank* model. Particularly, on CA_N dataset, the response times of four algorithms (UQuadTKIS, RTKIS, Naive and UTKIS) are 17.8, 44.28, 84 and 140 seconds respectively. They are 20.14, 52.85, 103 and 166.2 seconds respectively

on RT_U dataset.

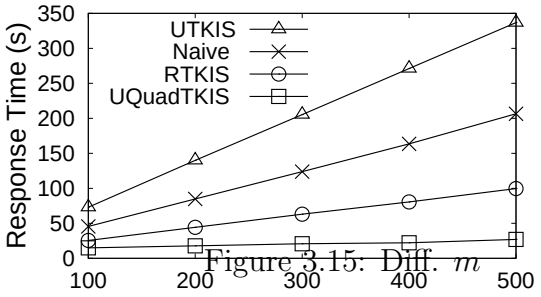


Figure 3.15: Diff. m

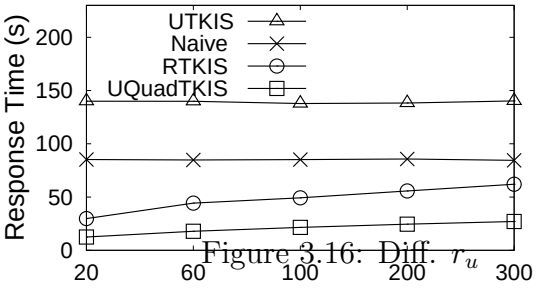


Figure 3.16: Diff. r_u

Impact of the number of instances (m). We evaluate the response time of the algorithms as a function of the number of instances (m) in each uncertain object which varies from 100 to 500. Clearly, the refinement cost increases in *refinement* phase when m grows. Figure 3.15 shows that UQuadTKIS has the best scalability against m, followed by RTKIS, Naive and UTKIS.

Impact of the radius(r_u). Figure 3.16 investigates the performance of four algorithms as a function of the radius size which varies from 20 to 300. It is shown that the scalability of UQuadTKIS is better than that of RTKIS regarding the growth of r_u.

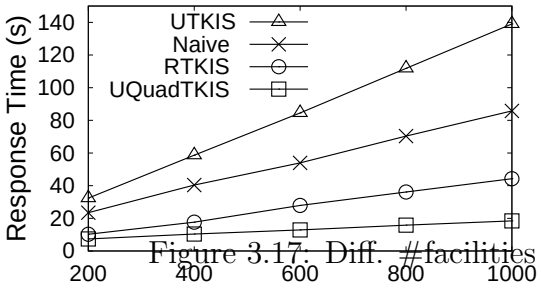


Figure 3.17: Diff. #facilities

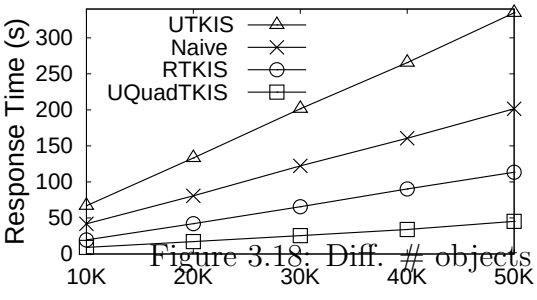


Figure 3.18: Diff. # objects

Impact of the number of facilities and objects. We also evaluate the impact of the number of facilities as well as the number of objects against four algorithms, where the number of facilities grows from 200 to 1000, and the number of objects

varies from 10,000 to 50,000. Figure 3.17 and Figure 3.18 show that UQuadTKIS has the best scalability among four algorithms.

Impact of k . In the last set of experiments, we evaluate the response time and the number of I/O accesses against various k values in Figure 3.19, which indicates that the performance of all algorithms are not very sensitive to the k value.

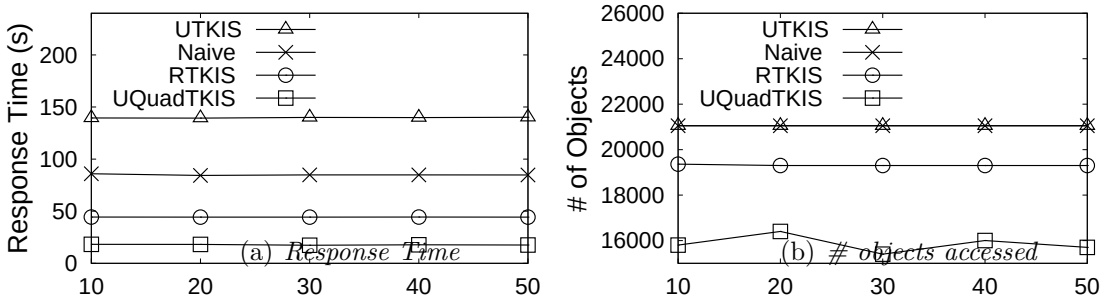


Figure 3.19: Diff. k

3.4.3 Randomized Algorithms Performance Evaluation

We will evaluate the performance of three randomized algorithms (Naive*, RTKIS*, UQuadTKIS*) in this sub-section.

Evaluating Accuracy

We evaluate the accuracy of randomized algorithms by *precision* and *recall* of the top k results. As we output exactly k facilities for top k query, *precision* and *recall* of the results are always the same.

Figure 3.20 reports the impact of the number of sample sets s on randomized algorithms on default dataset. As expected, the approximation quality improves when s grows. With only 200 rounds of trials, the randomized algorithms can achieve a *precision* and *recall* values of at least 0.97 for all algorithms. Therefore,

s is set to 200 in the following experiments.

we also evaluate the impact of k on randomized algorithms in Figure 3.21. Although the accuracy slightly decreases when k grows, *precision* and *recall* values are higher than 0.97.

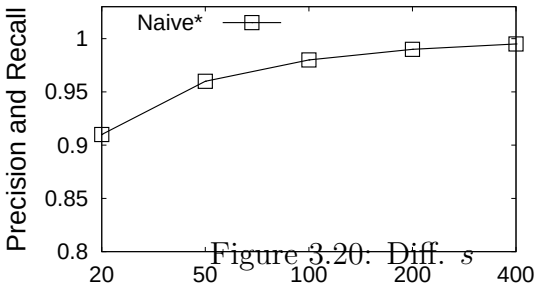


Figure 3.20: Diff. s

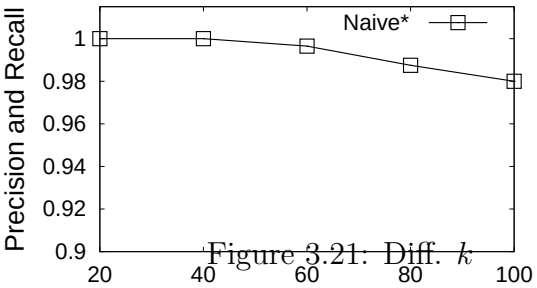


Figure 3.21: Diff. k

Evaluating Efficiency

In the first experiment, we vary the value of k and evaluate the performance of the five algorithms (we exclude Naive due to its poor performance in Section 3.4.2) proposed in the chapter against the default datasets where k varies from 10 to 50 in Figure 3.22. As expected, all techniques proposed are effective and not sensitive to various k values. For better report on the performance of the algorithms, we exclude the RTKIS algorithm in the following experiments since it has been significantly outperformed by the other algorithms. Figure 3.22 shows that UQuadTKIS* has the best performance and scalability among all algorithms.

Impact of the number of instances (m). We evaluate the response time of the algorithms as a function of the number of instances (m) in each uncertain object which varies from 500 to 4K. Figure 3.15 shows that the cost of UQuadTKIS increases dramatically because the cost of exact algorithm UQuadTKIS increases in *refinement* phase when m grows. As expected, the performances of randomized algorithms are not affected by m, because we only obtain 200 sample sets to

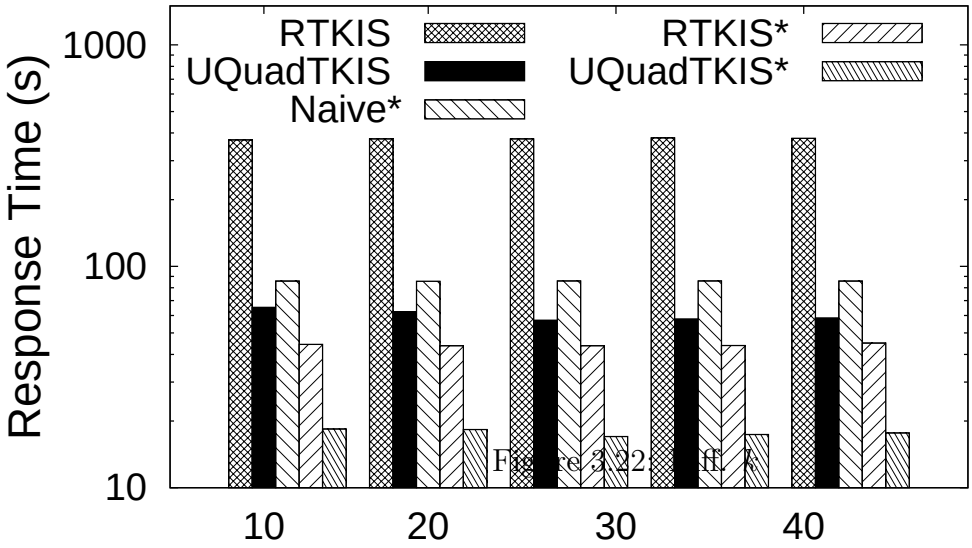


Figure 3.22: Effect of k

perform the queries regardless of the number of instances m. This implies that the randomized algorithm can efficiently handle the massive number of instances for uncertain objects. Moreover, it shows that UQuadTKIS* always has the best performance.

Impact of the radius (r_u). Figure 3.24 investigates the performance of four algorithms as a function of the radius size which various from 20 to 300. The response time of UQuadTKIS rises rapidly when r_u increases, and the Naive* algorithm is not sensitive to r_u. The costs of UQuadTKIS* and RTKIS* increase slowly when r_u grows. It is shown that the UQuadTKIS* outperforms RTKIS* and Naive* under all settings of r_u.

Impact of the number of facilities and objects. We also evaluate the impact of the number of facilities as well as the number of objects against four algorithms, where the number of facilities grows from 200 to 1000, and the number of objects varies from 10K to 50K. Figure 3.17 and Figure 3.18 show that UQuadTKIS* has the best scalability among four algorithms.

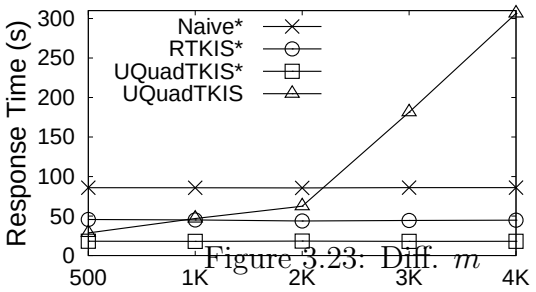


Figure 3.23: Diff. m

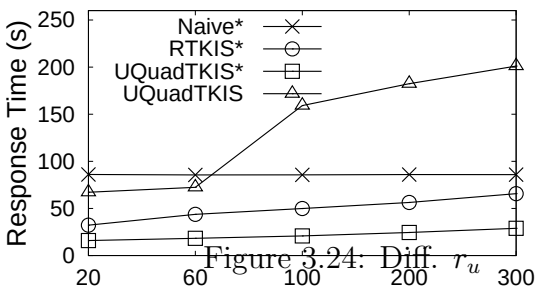


Figure 3.24: Diff. r_u

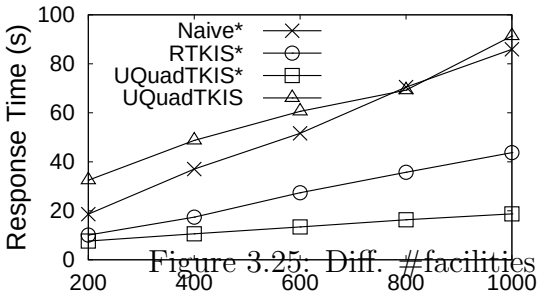


Figure 3.25: Diff. #facilities

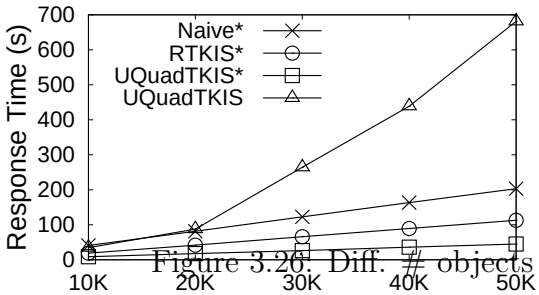


Figure 3.26: Diff. # objects

Impact of the sample set size (s). In the last experiments, we evaluate the performance of three randomized algorithms against the number of sample sets s which varies from 20 to 400. As expected, Figure 3.27 shows that the time cost of three algorithms increases with s. Among three algorithms, UQuadTKIS* always outperforms the other two competitors, and it also has the best scalability regarding the growth of s.

3.4.4 Summary

As a short summary, our comprehensive performance study shows that our ranking model has very similar ranking result with that of *expected rank* model, while the efficiency of the algorithms under our ranking model is much better. Even a naive implementation can outperform UTKIS Algorithm which follows the *expected rank* model. The experiments also show the effectiveness and efficiency of

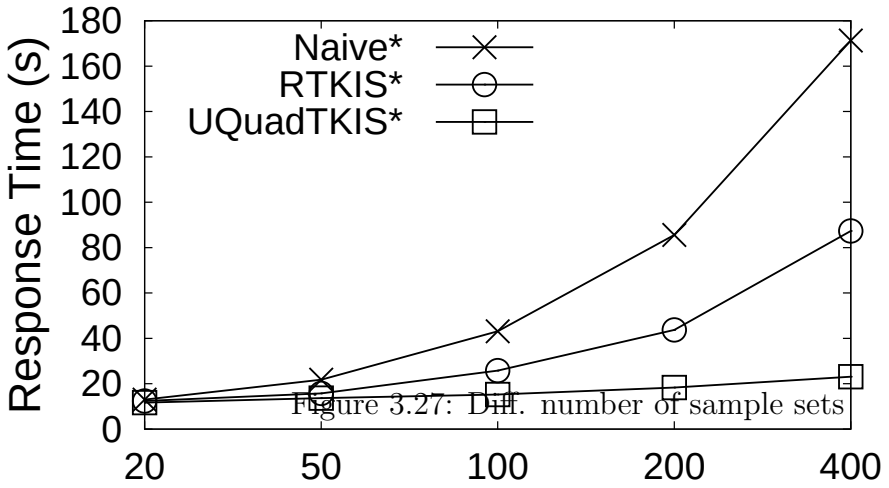


Figure 3.27: Diff. number of sample sets

the *filtering* and *refinement* algorithms proposed in the chapter based on R-tree and U-Quadtree. The overall performance of the U-Quadtree based algorithm (UQuadTKIS) always outperforms the R-tree based one (RTKIS) under various experiment settings because more sophisticated indexing structure is employed in UQuadTKIS. When a large number of instances per object are involved into the query, the randomized technique can significantly reduce the cost with the *precision* and *recall* are above 0.97. As expected, the overall performance of the UQuadTKIS* are always the best.

3.5 Conclusion

In this chapter, we investigate the problem of finding top k most influential facilities over a set of uncertain objects. Based on a new ranking semantics, we develop effective and efficient algorithms by utilizing two uncertain objects indexing techniques, R-tree and U-Quadtree respectively. A set of pruning techniques are proposed in this chapter to significantly improve the performance of the *filtering* and *refinement* algorithms. We further develop efficient randomized algorithms with

accuracy guarantee to tackle uncertain objects with massive number of instances. Our experiments convincingly demonstrate the effectiveness and efficiency of our techniques.

Chapter 4

Identify Top k Dominating Objects

A straightforward solution for *parameterized ranking* based top k dominating query is to first calculate the *dominance score* for each instance of the objects, and then apply the *parameterized ranking* algorithm [LSD11] to identify the top k dominating objects. However, it is cost-inhibitive because the cost of *dominance score* computation is not cheap and the total number of instances may be huge. Therefore, we propose an effective and efficient algorithm to support the top k dominating query by developing novel pruning techniques based on popular R-tree based indexing structure and some simple statistics information.

This chapter is organized as follows. Section 4.1 introduces the problem and some preliminary knowledge. Section 4.2 develops efficient algorithms to support the top k dominating query by utilizing the spatial indexing and statistics information. The experimental results are reported in Section 4.3. We conclude the chapter in Section 4.4.

4.1 Preliminary

In this section, we first formally introduce the multi-dimensional uncertain object model and the *parameterized ranking* function, as well as the problem statement. Section 4.1.2 illustrates how to calculate the *rank scores* of the uncertain objects based on the *generating function*. Section 4.1.3 introduces the R-tree based indexing structure for uncertain objects. We present problem definition and necessary preliminaries in this section. Table 4.1 summarizes notations frequently used throughout the chapter.

Notation	Meaning
U, V	uncertain objects
$\mathcal{O}$	a set of uncertain objects
n	the number of uncertain objects in $\mathcal{O}$
u, v	instances of the uncertain objects
m	the number of instances in each uncertain object
$u \prec v$	u *dominates* v
d	dimensionality of the space
p	a point (instance) in a d-dimensional space
$p.D_i$	i-th coordinate value of point p
p_u	occurrence probability of the instance u
$s(u)$	*dominance score* of the instance u
PRF^{ω}	*parameterized ranking* function
$\Upsilon(U)/\Upsilon(u)$	the *rank score* of an object U/instance u
$\Upsilon(u)$	the *rank score* of an instance u
U_{mbr}	minimal bounding rectangle of U
U^-_{mbr} (U^+_{mbr})	lower (upper) corner point of U_{mbr}
U_s	*dominance score* distribution of U
U^-_s (U^+_s)	lower (upper) bound of the *dominance scores* for instances in U
$P(x, U)$	probability mass of the instances of U, which are dominated by point (instance) x

Table 4.1: The summary of notations

4.1.1 Problem Definition

A point (instance) p is in a d-dimensional space and the i-th dimensional coordinate value of p is denoted by $p.D_i$. Without loss of generality, we assume smaller coordinate values are preferred. For two points p and q, p *dominates* q, denoted by $p \prec q$, if $p.D_i \leq q.D_i$ for all dimension $i \in [1, d]$ and there is at least one dimension $j \in [1, d]$ with $p.D_j < q.D_j$. Meanwhile, we use $p \preceq q$ to denote that p *dominates or equals* q.

Uncertain Object Model. An uncertain object can be described either *continuously* or *discretely*. In the chapter, we focus on the *discrete* case. Note that we can discretize a continuous probability density function (PDF) of an uncertain object by sampling methods. In the *discrete* case, an uncertain object U consists of a set $\{u_1, u_2, \ldots, u_m\}$ of instances (points). For $1 \leq i \leq m$, an instance u_i occurs with probability p_{u_i} ($p_{u_i} > 0$), and $\sum_{i=1}^{m} p_{u_i} = 1$. We assume that the uncertain objects are *independent* to each other. In the following chapter, we use *object* to denote *multi-dimensional uncertain object* whenever there is no ambiguity. Given an object U, U_{mbr} denotes the minimal bounding rectangle which contains all of the instances of U. Let U^-_{mbr} (U^+_{mbr}) denote the lower (upper) corner of U_{mbr}, we have $U^-_{mbr} \preceq u$ and $u \preceq U^+_{mbr}$ for any instance $u \in U_{mbr}$.

Dominance Score Distribution. Based on the dominance relationship against two points (instances), we can easily measure the goodness of an instance as follows.

Definition 4.1 (Dominance Score)**.** *Given a set $\mathcal{O}$ of objects, the dominance score of an instance u of the object U, denoted by $s(u)$, is*

$$s(u) = \sum\nolimits_{V \in\ \mathcal{O} \setminus U}\ \sum\nolimits_{v \in V\ \wedge\ u \prec v} p_v \tag{4.1}$$

where $\sum_{v \in V \wedge u \prec v} p_v$ represents the probability that the object V is dominated by

the instance u.

Given a set $\mathcal{O}$ of objects, we can derive the *dominance score distribution* for each object $U \in \mathcal{O}$, denoted by U_s, where $U_s = \{(s(u_1), p_{u_1}), \ldots, (s(u_i), p_{u_i}), \ldots, (s(u_m), p_{u_m})\}$. We use $Pr(U_s > c)$ to denote the probability that U_s is larger than the value c, where $Pr(U_s > c) = \sum_{u_i \in U \wedge s(u_i) > c} p_{u_i}$.

Example 4.1. *Regarding the example in Figure 1.2(b) and Definition 4.1, dominance score distributions of four objects A, B ,C and D are depicted in Figure 4.1 where we assume each instance has appearance probability 0.5. Particularly, as a_1 dominates four instances (b_1, b_2, d_2, and c_2), and a_2 only dominates c_2, we have $s(a_1) = 2$, $s(a_2) = 0.5$. Since $p_{a_1} = p_{a_2} = 0.5$, we get $A_s = \{(2, 0.5), (0.5, 0.5)\}$. Similarly, we have $B_s = \{(1, 0.5), (0.5, 0.5)\}$, $C_s = \{(0.5, 0.5), (0, 0.5)\}$, $D_s = \{(2.5, 0.5), (1.5, 0.5)\}$, and $Pr(D_s > 2) = p_{d_1} = 0.5$.*

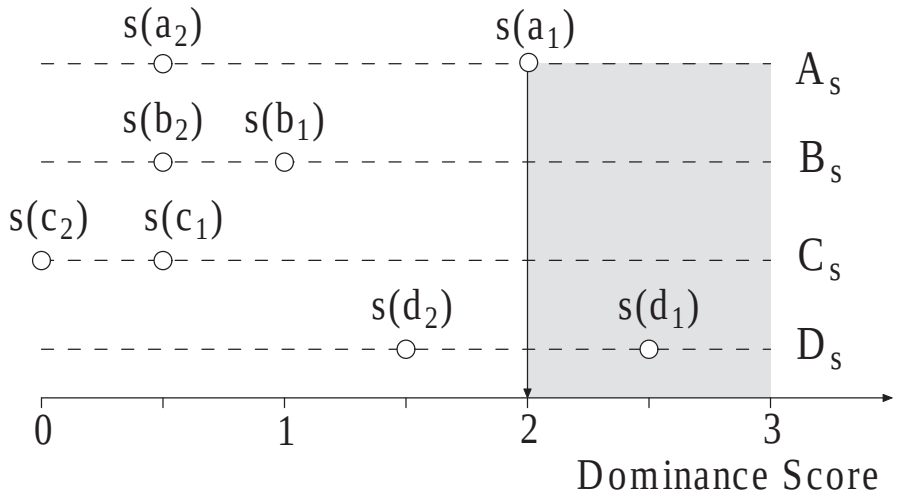


Figure 4.1: Dominance Score Distributions

Parameterized Ranking. The *dominance score distributions* of a set of objects can be ranked by the top k semantics studied for uncertain data in the literature. In the chapter, we focus on the *parameterized ranking* function (PRF^{ω}) proposed in [LSD11] since it can unify other popular ranking functions.

For a set $\mathcal{O}$ of objects, a possible world W proposed in [DS07] is a set of instances with one instance from each uncertain object. Given a set of uncertain objects $\{U_1, U_2, \ldots, U_n\}$, a possible world $W = \{u_1, u_2, \ldots, u_n\}$ is a set of instances sequentially sampled from each object. Assume the uncertain objects are independent to each other, and the probability of W to appear is $Pr(W) = \prod_{i=1}^{n} p_{u_i}$. In each world W, an object is ranked based on the score (value) of its corresponding instance in W. In the chapter, we use $r_W(U)$ to denote the rank of an object in the possible world W which is abbreviated to $r(U)$ whenever there is no ambiguity. Let $\mathcal{W}$ denote the set of all possible worlds, and we have $\sum_{W \in \mathcal{W}} Pr(W) = 1$ where $Pr(W)$ is the occurring probability of the possible world W. Then we have the formal definition of *parameterized ranking* function.

Definition 4.2 (PRF^{ω})**.** *Let ω be a weighted function which maps an object-rank pair to a complex number, the* ***rank score*** *of an object U, denoted by $\Upsilon(U)$, is defined as follows.*

$$\Upsilon(U) = \sum\nolimits_{i \in [1,n]} \omega(i) \times Pr(r(U) = i) \tag{4.2}$$

where $\omega(i)$ denotes the weight of the i-th position, and $Pr(r(U) = i)$ denotes the probability of U ranked at the i-th position, i.e., $Pr(r(U) = i) = \sum_{W \in \mathcal{W} \wedge r_W(U) = i} Pr(W)$. Recall that $r(U)$ denotes the rank of U in the possible world W.

In the chapter, we assume $w(i) \geq w(j)$ for any two ranking positions i and j where $i < j$. This is very intuitive as a higher position is usually at least as desirable as those behind it and thus should be given a higher weight.

Problem Statement. Given a set $\mathcal{O}$ of objects, we aim to return the top k dominating objects based on their *dominance score distributions* and *parameterized*

ranking semantics; that is, we retrieve the top k objects with the highest *rank scores* regarding their *dominance score distributions*.

4.1.2 Computing Rank Score $\Upsilon(U)$

As shown in [LSD11], the *rank score* of an object U can be calculated by the summation of the *rank scores* of its instances. For an instance $u \in U$, we can use the following *generating function* $\mathcal{F}(\mathbf{x}, u)$ to calculate the *rank score* of u, where $P_{V,u}$ is the probability that V_s is larger than $s(u)$ (i.e., $P_{V,u} = Pr(V_s > s(u)) = \sum_{v \in V \wedge s(v) > s(u)} p_v$). Recall that V_s is the *dominance score distribution* of the object V. Intuitively, a small $P_{V,u}$ is in favor of the *rank score* of the instance u.

$$\mathcal{F}(\mathbf{x}, u) = \left(\prod\nolimits_{V \in \mathcal{O} \setminus U} (1 - P_{V,u} + P_{V,u} \cdot \mathbf{x}) \right) \ (p_u \cdot \mathbf{x}) \tag{4.3}$$

Then we have $Pr(r(u) = i) = c_i$ where $r(u)$ is the rank position of instance u and c_i is the coefficient of $\mathbf{x}^i$ in $\mathcal{F}(\mathbf{x}, u)$. Therefore, after applying Equation 4.2, we have

$$\Upsilon(u) \ = \ \sum\nolimits_{1 \le i \le n} \omega(i) \times c_i \tag{4.4}$$

and

$$\Upsilon(U) \ = \ \sum\nolimits_{u \in U} \Upsilon(u) \tag{4.5}$$

Example 4.2. *In Figure 4.1, we have* $s(a_1) = 2$ *and hence* $Pr(B_s > s(a_1)) = 0.0$, $Pr(C_s > s(a_1)) = 0.0$, *and* $Pr(D_s > s(a_1)) = 0.5$ *(i.e., the probability mass of the instances of* D *in the shaded area). According to Equation 4.3,* $\mathcal{F}(\mathbf{x}, a_1) = (1) \times (1) \times (0.5 + 0.5\ \mathbf{x}) \times 0.5\ \mathbf{x} = 0.25\ \mathbf{x} + 0.25\ \mathbf{x}^2$. *Therefore, we have* $Pr(s(a_1) = 1) = 0.25$ *and* $Pr(s(a_1) = 2) = 0.25$. *Similarly,* $\mathcal{F}(\ \mathbf{x}, a_2) = (0.5 + 0.5\ \mathbf{x}) \times (1) \times (\mathbf{x}) \times 0.5\ \mathbf{x} = 0.25\ \mathbf{x}^2 + 0.25\ \mathbf{x}^3$, *and hence* $Pr(s(a_2) = 1) = 0$, $Pr(s(a_2) = 2) = 0.25$ *,and* $Pr(s(a_2) = 3) = 0.25$. *Suppose* $\omega(1) = 4$, $\omega(2) = 3$, $\omega(3) = 2$ *and* $\omega(4) = 1$ *in*

PRF^ω, then $\Upsilon(A) = \Upsilon(a_1) + \Upsilon(a_2) = (0.25 \times 4 + 0.25 \times 3) + (0.25 \times 3 + 0.25 \times 2) = 3$ *according to Equation 4.4 and 4.5. Similarly, we have* $\Upsilon(B) = 2.5$, $\Upsilon(C) = 1.5$ *and* $\Upsilon(D) = 3.75$. *Therefore,* $\Upsilon(D) > \Upsilon(A) > \Upsilon(B) > \Upsilon(C)$.

Algorithm 7: ***Compute Rank Scores***$(\mathcal{O}, PRF^\omega)$

Input : $\mathcal{O}$: a set of objects,

PRF^ω : parameterized ranking function,

Output: $\mathcal{S}$: top k objects with highest *rank scores*

1 Sort instances of objects in decreasing order regarding their *dominance scores*;

2 $\mathcal{G}^0(\mathbf{x}) := 1$; $acct(U) := 0$;

3 **for** *each* t_i *accessed in order* **do**

4 $P_{old} := acc(U^{t_i})$;

5 $\mathcal{F}(\mathbf{x}, t_i) := \frac{\mathcal{G}^{i-1}(\mathbf{x})}{1 - P_{old} + P_{old}\ \mathbf{x}} p_{t_i}\mathbf{x}$;

6 Compute $\Upsilon(t_i)$ based on $\mathcal{F}(\mathbf{x}, t_i)$;

7 $acc(U^{t_i}) := acc(U^{t_i}) + p_{t_i}$;

8 $P_{new} := acc(U^{t_i})$;

9 $\mathcal{G}^i(\mathbf{x}) := \mathcal{G}^{i-1}(\mathbf{x}) \frac{1 - P_{new} + P_{new}\ \mathbf{x}}{1 - P_{old} + P_{old}\ \mathbf{x}}$;

Suppose the *dominance scores* of the instances are available and the instances of the objects are ordered by their *dominance scores* in decreasing order, it takes $\mathcal{O}(\ m^3n^3)$ time to compute the *rank scores* of the instances following Equation 4.3, where n and m represent the number of objects and the average number of instances per object. In [LSD11], the *parameterized ranking* based top k algorithm for attribute level uncertain objects (e.g., uncertain objects with multiple instances) is complicate since the correlation of the uncertain objects are involved. Algorithm 7 illustrates a simple version of the *parameterized ranking* based top k algorithm with

independent assumption among objects, and the time complexity is $\mathcal{O}(m^2n^2)$. The key idea is to incrementally maintain a *generating function* to facilitate the *rank scores* computation of the instances, which are accessed in decreasing order of their *dominance scores*.

Suppose $t_1, t_2, .. t_l$ are instances ordered by their *dominance scores* in decreasing order [1], and $s(t_i)$ denotes the *dominance score* of the tuple t_i. Let $\mathcal{G}^i(\mathbf{x})$ denote the expansion of the *generating function* regarding the instance t_i for any object V in the set $\mathcal{O}$, where

$$\mathcal{G}^i(\mathbf{x}) = \prod_{V \in \mathcal{O}} (1 - Pr(V_s \geq s(t_i)) + Pr(V_s \geq s(t_i)) \cdot \mathbf{x}) \tag{4.6}$$

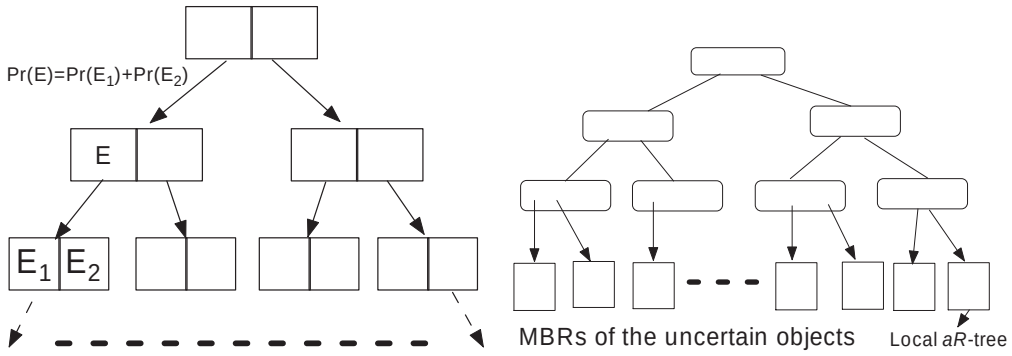
Let U^{t_i} denote the object which contains the instance t_i, then we have $\mathcal{G}^i(\mathbf{x}) = \mathcal{G}^{i-1}(\mathbf{x}) \frac{P_i(U^{t_i}, \mathbf{x})}{P_{i-1}(U^{t_i}, \mathbf{x})}$ where $P_i(U^{t_i}, \mathbf{x}) = 1 - Pr(U_s^{t_i} \geq s(t_i)) + Pr(U_s^{t_i} \geq s(t_i))\mathbf{x}$ and $P_{i-1}(U^{t_i}, \mathbf{x}) = 1 - Pr(U_s^{t_i} \geq s(t_{i-1})) + Pr(U_s^{t_i} \geq s(t_{i-1}))\mathbf{x} = 1 - Pr(U_s^{t_i} > s(t_i)) + Pr(U_s^{t_i} > s(t_i))\mathbf{x}$. Based on Equation 4.6, we have $\mathcal{F}(\mathbf{x}, t_i) = \frac{\mathcal{G}^{i-1}(\mathbf{x})}{P_{i-1}(U^{t_i}, \mathbf{x})} p_{t_i} \mathbf{x}$. Then we initialize $\mathcal{G}^0(\mathbf{x}) = 1$, consequently $\mathcal{F}(\mathbf{x}, t_1) = p_{t_1}\mathbf{x}$, note that $P_0(U^{t_1}, \mathbf{x}) = 1 - Pr(U_s^{t_1} \geq s(t_0)) + Pr(U_s^{t_1} \geq s(t_0))\mathbf{x} = 1$ since the t_0 does not exist. Algorithm 7 illustrates the details of the computation where the *rank score* of an instance can be computed based on the *generating function* $\mathcal{G}(\mathbf{x})$. In Algorithm 7, for each object U we use $acc(U)$ to record the probability mass of its instances seen so far.

4.1.3 Indexing Uncertain Objects by R-tree

To facilitate the top k dominating query, we assume uncertain objects are organized by R-tree indexing techniques. Given a set $\mathcal{O}$ of uncertain objects, Figure 4.2 illustrates the basic idea of the R-tree based indexing approach where the MBRs of the objects are indexed by R-tree [Gut84], which is called the *global* R-tree of

[1] For presentation simplicity, we assume the *dominance scores* of the instances are unique.

$\mathcal{O}$. As to each uncertain object U, an *aggregate* R-tree [PKZT01] is employed to organize the instances where the aggregate value of each intermediate entry is the probability mass of the instances in the entry, which is called a *local* R-tree for an uncertain object U.



(a) *Local aR-tree* (b) *Global R-tree*

Figure 4.2: R-tree based Indexing

4.2 Approach

A straightforward solution for the problem of top k dominating query is to first calculate the *dominance score* for each instance of the objects by conducting dominance checks against the instances of other objects, then compute the *rank scores* of the objects based on Algorithm 7 in Section 4.1.2. The k objects with the highest *rank scores* are the top k dominating objects. However, it is cost-inhibitive because the cost of *dominance score* computation is not cheap and the total number of instances may be huge. In this section, we propose efficient algorithms to identify the top k dominating uncertain objects. Specifically, Section 4.2.1 presents the framework of our algorithms following the *filtering and verification* paradigm. Section 4.2.2 proposes efficient algorithms for the computation of the *dominance score*. Section 4.2.3 introduces the spatial pruning technique based on the MBRs

of the objects. Rank score based pruning technique is proposed in Section 4.2.4. In Section 4.2.5, statistics based approach further improves the performance of the algorithms by utilizing the probabilistic inequalities.

To facilitate the top k dominating query, we assume uncertain objects are organized by R-tree indexing techniques. Given a set $\mathcal{O}$ of uncertain objects, Figure 4.2 illustrates the basic idea of the R-tree based indexing approach where the MBRs of the objects are indexed by R-tree [Gut84], which is called the *global* R-tree.As to each uncertain object U, an *aggregate* R-tree [PKZT01] is employed to organize the instances where the aggregate value of each intermediate entry is the probability mass of the instances in the entry, which is called a *local* R-tree for an uncertain object U.

4.2.1 Compute Top k Dominating Objects

In this subsection, we introduce the framework of the top k dominating algorithm based on the *filtering and verification* paradigm in Algorithm 8. Pruning techniques are developed to reduce computational cost by eliminating non-promising objects. Assume MBRs of the objects are organized by the *global* R-tree $\mathcal{R}$, Line 2 derives the lower and upper bounds of the *dominance scores* for each object U, denoted by U_s^- and U_s^+ respectively (See Section 4.2.2), ,where $U_s^- \leq s(u) \leq U_s^+$ for any instance $u \in U$. Then Line 3 applies the spatial based pruning technique to identify the candidate objects which are kept in a set $\mathcal{C}$. In addition to the candidate objects, $\mathcal{L}$ keeps the objects whose instances may contribute to the *rank scores* computation of candidate objects. As shown in Section 4.2.3, we can safely remove remaining objects (i.e., $\mathcal{O} \setminus \mathcal{L}$) from *rank scores* computation [2]. A max-heap $\mathcal{H}$ is employed to guarantee that instances are accessed by decreasing order, which is initialized

[2]These objects may be involved in the *dominance score* computation of other objects.

by objects in $\mathcal{L}$ where the upper bounds of their *dominance scores* are key values in the heap(Line 5). Similar to Algorithm 7, the *generating function* $\mathcal{F}(x, u)$ is maintained to compute the *rank scores* for instances accessed (Line 9 and 11). Algorithm 8 maintains a *rank score* threshold λ to prune non-promising objects in Line 12 and 18 by utilizing *rank score* based pruning techniques (See details in Section 4.2.4). Line 19 loads the instances of an object and calculate their *dominance scores*, where an efficient *dominance scores* computation algorithm is presented in Section 4.2.2. Then Line 20 pushes these instances into the heap $\mathcal{H}$ for further processing. Line 6 terminates the algorithm when there is no candidate object for further exploration or the heap is empty. Finally the k objects with the highest *rank scores* are returned as the top k dominating objects.

4.2.2 Compute Dominance Scores

In this subsection, we introduce efficient *dominance scores* computation algorithms based on the R-tree structure. Specifically, we first introduce the *dominance relationship* between two rectangles, and then show how to calculate the lower and upper bounds of the *dominance scores* for the instances of an object based on their MBRs, which are organized by *global* R-tree. Then the algorithm is also extended to support computation of *dominance scores* for all instances of an object.

Following is a formal definition of the *dominate relationship* between two rectangles.

Dominance Relationships for Rectangles. A pair of uncertain objects may have three relationships as follows. Let R^+ and R^- denote the upper and lower corners of a rectangle R, we say the rectangle R_1 *fully dominates* another rectangle R_2 if $R_1^+ \prec R_2^-$. Similarly, we have R_1 *partially dominates* R_2 if $R_1^- \prec R_2^+$ and $R_1^+ \not\prec R_2^-$. Otherwise, we say R_1 does not dominate R_2. It is immediate that we

Algorithm 8: ***Top k Dominating Objects***($\mathcal{R}$, k)

Input : $\mathcal{R}$: the *global* R-tree of $\mathcal{O}$,

k: objects retrieved

Output: Top k dominating objects

```
1  λ := 0;
2  Compute U_s^+ and U_s^- for all objects U ∈ O;
3  L, C ← spatial based pruning against O;
4  for each U ∈ L do
5      Push U into H with key value U_s^+;
6  while H ≠ ∅ or C ≠ ∅ do
7      E ← deheap(H);
8      if E is an instance u from object U then
9          Update the generating function F(x, u);
10         if U ∈ C then
11             Compute Υ(u) based on F(x, u);
12             C := C \ U if U can be pruned by λ;
13             if u is the last instance of U then
14                 Compute Υ(U); Update λ; C := C \ U;
15     else
16         E corresponds to the object U;
17         if U can be pruned based on λ then
18             C := C \ U;
19         Compute dominance scores for instances of U;
20         Push every u ∈ U into H with key value s(u);
21 return Top k objects with highest rank scores
```

have $x \prec y$ for any points $x \in R_1$ and $y \in R_2$ if R_1 *fully dominates* R_2. Similarly, $x \not\prec y$ for any points $x \in R_1$ and $y \in R_2$ if R_1 does not *dominate* R_2.

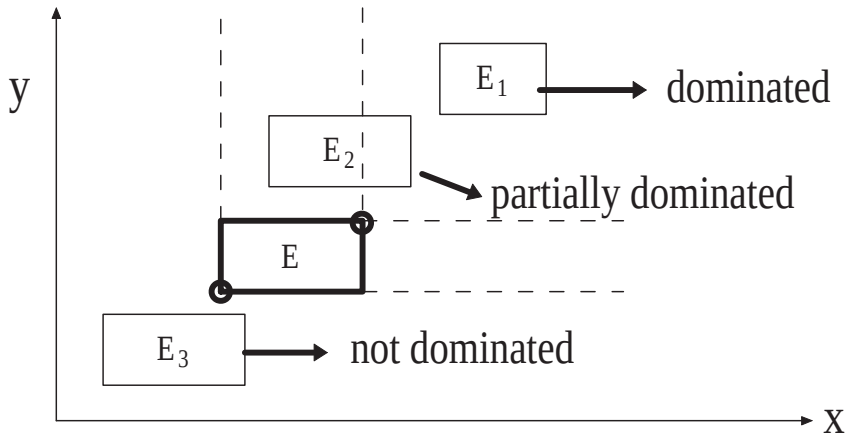


Figure 4.3: MBR Dominance Relationships

Example 4.3. *As shown in Figure 4.3, we have E fully dominates E_1, E partially dominates E_2 and E does not dominate E_3.*

Compute the Lower and Upper Bounds (U_s^-, U_s^+)

Based on the definition of the *dominance score* (Definition 4.1) and the *dominance relationship* between two rectangles, we can derive the lower and upper bounds of the *dominance scores* of the instances from an object U, denoted by U_s^+ and U_s^- respectively, based on the MBRs of the objects. In this subsection, U_s^- equals the number of other objects whose MBRs are *fully dominated* by U_{mbr}. Similarly, U_s^+ is the number of other objects whose MBRs are *fully dominated* or *partially dominated* by U_{mbr}. In Section 4.2.5, these bounds can be improved by utilizing statistics information of the objects.

Motivation. We may issue two dominating range queries based on the lower and upper corners of the MBR of each object against the *global* R-tree $\mathcal{R}$ in Algorithm 8 to compute their lower and upper bounds of the *dominance scores*. Nevertheless,

we can improve the computational cost by conducting a spatial join based computation. The main idea is that, instead of directly computing *dominance score* for each individual instance, we conduct the dominance checks in a level-by-level fashion such that the dominance count can be calculated at higher level, and hence significantly reduce the number of dominance checks.

Algorithm. Algorithm 9 illustrates the details of the *dominance score* bounds computation based on MBRs of the objects, which follows the *synchronized* R-tree traversal paradigm used in spatial join. For each entry E (data entry or intermediate entry) of the *global* R-tree $\mathcal{R}$, we use a tuple T to record its lower and upper bounds of its *dominance score*, denoted by T_s^- and T_s^+ respectively, while $T.owner$ refers to the entry E. Clearly, we do not need to further explore another entry E_1 regarding E if E *fully dominates* E_1 or E does not *dominate* E_1. We use $T.set$ to keep a set of entries which are *partially dominated* by E. A FIFO queue $\mathcal{Q}$ is employed to maintain the tuples, and $\mathcal{Q}$ is initialized by a tuple T where $T.owner$ and $T.set$ are set to the root of the *global* R-tree $\mathcal{R}$ (Line 1-2). For each tuple T popped from $\mathcal{Q}$, if the entries from $T.owner$ and $T.set$ are data entries, then we have $U_s^+ = T_s^+$ and $U_s^- = T_s^-$ where $T.owner$ refers to the MBR of the object U. Otherwise, Line 7-17 expand $T.owner$ and $T.set$ for *dominance score* computation of the lower level entries. For presentation simplicity, the child entries of a data entry are referred to the data entry itself (Line 7 and 11). Specifically, for each pair of entries $t.owner$ and e at Line 12 and 13, the lower bound t_s^- and upper bound t_s^+ are increased by $e.cn$ if t_{mbr} *fully dominates* e_{mbr}, where $e.cn$ is the aggregate number of objects in entry e (Line 15). Otherwise, we only increase t_s^+ and keep e in $t.set$ for further computation if t_{mbr} *partially dominates* e_{mbr} (Line 17). Algorithm 9 terminates when $\mathcal{Q}$ is empty, and the lower and upper *dominance score* bounds of the objects are ready for the spatial pruning in Section 4.2.3.

Algorithm 9: ***Compute Dominance Score Bounds*** $(\mathcal{R})$

Input : $\mathcal{R}$: the *global* R-tree of $\mathcal{O}$

Output: Objects with lower and upper *dominance scores* bounds

```
1  T.owner := root of R ; T.set := root of R;
2  Push T into FIFO queue Q;
3  while Q ≠ ∅ do
4      T ← dequeue(Q);
5      if T.owner is not a data entry or entries in T.set are not data entries
       then
6          L := ∅; W := ∅;
7          for each child entry e of T.owner do
8              t.owner = e; t_s^- := T_s^-; t_s^+ := t_s^-;
9              L := L ∪ t;
10         for each entry e in T.set do
11             W := W ∪ child entries of e;
12         for each tuple t in L do
13             for each entry e in W do
14                 if t_mbr fully dominates e_mbr then
15                     t_s^- := t_s^- + e.cn; t_s^+ := t_s^+ + e.cn;
16                 else if t_mbr partially dominates e_mbr then
17                     t_s^+ := t_s^+ + e.cn ; t.set := t.set ∪ e;
18             Push t into Q;
```

Compute the Dominance Score

In the top k dominating algorithm (Line 19 of Algorithm 8) we need to compute the *dominance scores* of the instances for the object U. We can come up with an efficient *dominance scores* computation algorithm for an object U by slightly modifying Algorithm 9. Suppose U_s^- is calculated, we only need to consider a set $\mathcal{S}$ of objects which are *partially dominated* by U. For each object $V \in \mathcal{S}$, we set $T.owner$ and $T.set$ to the roots of R_U and R_V respectively at Line 1 where R_U and R_V are the local R-tree of the objects U and V respectively. When algorithm terminates, for each instance $u \in U$ we have the probability mass of instances from V which are dominated by u, denoted by $P(u, V)$. According to Definition 4.1, the *dominance score* of the instance u can be calculated as follows.

$$s(u) = U_s^- + \sum\nolimits_{V \in \mathcal{S}} P(u, V). \tag{4.7}$$

4.2.3 Spatial Pruning Technique

Considering that the cost is expensive to compute *dominance scores*, we propose effective spatial pruning techniques to reduce the number of candidate objects based on the *global* R-tree; that is, we aim to prune a set of objects from *dominance score* computation without accessing the instances of the objects such that the CPU and I/O costs can be significantly reduced.

Following theorem indicates that an object U is ranked higher than another object V if the lower *dominance score* bound of U is larger than the upper *dominance score* bound of V.

Theorem 4.1. *Given two objects U and V, we have $\Upsilon(U) > \Upsilon(V)$ if $U_s^- > V_s^+$.*

Proof. Since $U_s^- > V_s^+$, we have $s(u) > s(v)$ for any $u \in U$ and $v \in V$. Therefore,

we have $Pr(r(U) \leq i) > Pr(r(V) \leq i)$ where $Pr(r(U) \leq i)$ denotes the probability that U is ranked not lower than the i-th position. According to Equation 4.2 and the monotonic property of the weight function (i.e., $\omega(i) \geq \omega(j)$ for any two positions i and j with $i < j$), we have $\Upsilon(U) > \Upsilon(V)$. □

Spatial based Pruning. Let f_c denote the k-th largest lower bound of the *dominance scores* regarding objects in $\mathcal{O}$, according to Theorem 4.1, we have $\mathcal{C} = \{U | U \in \mathcal{O}$ and $U_s^+ \geq f_c\}$ in Algorithm 8; that is, only the object U with $U_s^+ \geq f_c$ can become the top k candidate objects. Let f_s denote the smallest lower bounds of *dominance scores* regarding objects in $\mathcal{C}$, we have $\mathcal{L} = \{U |\ U \in \mathcal{O}$ and $U_s^+ \geq f_s\}$ in Algorithm 8. Note that, as shown in Equation 4.3 the *rank score* of an instance can only be affected by other instances with larger *dominance scores*.

4.2.4 Rank Score based Pruning Technique

In this subsection, we propose effective pruning techniques based on the *rank scores* of the objects. Let λ in Algorithm 8 denote the k-th largest *rank scores* for objects accessed, clearly we can safely remove an object U from the top k candidates if we can claim the upper bound of $\Upsilon(U)$ is smaller than λ. In the following, we show how to derive an upper bound of $\Upsilon(U)$ in Algorithm 8.

Let $u_1, u_2, \ldots, u_m$ denote the instances of an object U which are sorted by their *dominance scores* in decreasing order. In Algorithm 8, these instances will be accessed in order. For a given instance u_i, we use U^i to denote a new object which is constructed by "pushing" instances after the i-th position to u_i; formally, we have $U^i = \{u_1^*, \ldots, u_m^*\}$,

$$U^i = \{u_1^*, \ldots, u_m^*\} \tag{4.8}$$

where $p_{u_j^*} = p_{u_j}$ for any $1 \leq j \leq m$, $s(u_j^*) = s(u_j)$ for $j \leq i$, and $s(u_j^*) = s(u_i)$ for

$i < j \leq m$.

Following theorem indicates that $\Upsilon(U^i) \geq \Upsilon(U)$ for any $1 \leq i \leq m$, where $\Upsilon(U^i)$ is the *rank score* of U^i when U is replaced by U^i.

Theorem 4.2. *Suppose an object U has m instances, then we have $\Upsilon(U^i) \geq \Upsilon(U)$ for any $1 \leq i \leq m$.*

Proof. For given i with $1 \leq i \leq m$, according to the definition of U^i, we have $s(u_j^*) \geq s(u_j)$ for $1 \leq j \leq m$, and hence $Pr(r(u_j^*) \leq t) \geq Pr(r(u_j) \leq t)$ for any position t. Recall that $r(u)$ is the rank position of u and $Pr(r(u) \leq t)$ is the probability that the instance u is ranked not lower than the t-th position. Consequently, for any two instances u_j^* and u_j with $1 \leq j \leq m$, we have $\sum_{1 \leq l \leq t} c_l^* \geq \sum_{1 \leq l \leq t} c_l$ where $c_l^* = Pr(r(u_j^*) = l)$ and $c_l = Pr(r(u_j) = l)$. Note that c_l^* and c_l represent the l-th coefficient of the *generating function* for u_j^* and u_j respectively. Since $\omega(t_1) \geq \omega(t_2)$ for any two positions t_1 and t_2 with $t_1 < t_2$, we have $\Upsilon(u_j^*) \geq \Upsilon(u_j)$ for any two instances $u_j^* \in U^i$ and $u_j \in U$ according to Equation 4.4. Based on Equation 4.5, we have $\Upsilon(U^i) \geq \Upsilon(U)$, and hence the theorem holds. □

Rank Score Based Pruning. In Algorithm 8, we invoke the *rank score* based pruning technique at Line 12 and 18. Let u_i be the i-th visited instance of U, we can calculate $\Upsilon(U)^+$ by setting $p_{u_i} = \sum_{i \leq j \leq m} u_j$ (i.e., move probability mass of the unvisited instances of U to u_i), and prune the object U from candidate set $\mathcal{C}$ at Line 12 if $\Upsilon(U)^+ \leq \lambda$. Recall that λ is the k-th largest *rank scores* of the objects seen so far. Let u_0 denote an instance where $s(u_0) = U_s^+$ and $p_{u_0} = 1.0$,i.e., an object constructed by pushing all instances of U to the lower corner of U_{mbr}. With similar rationale in Theorem 4.2, we have $\Upsilon(U^0) \geq \Upsilon(U)$. Consequently, at Line 18 of Algorithm 8, we can calculate $\Upsilon(U)^+$ based on U_s^+ without loading instances of U and remove U from candidate set $\mathcal{C}$ if $\Upsilon(U)^+ \leq \lambda$. It is immediate that we

can also remove any unvisited object $V \in \mathcal{C}$ (i.e., the object V with $V_s^+ < U_s^+$) in $\mathcal{C}$ from the top k candidates since we have $\Upsilon(V) \leq \Upsilon(V)^+$ and $\Upsilon(V)^+ \leq \Upsilon(U)^+$. In Algorithm 8, we invoke the *rank score* based pruning technique at Line 12 and 18. Let u_i be the i-th visited instance of U, we can calculate $\Upsilon(U^i)$ by setting $p_{u_i} = \sum_{i \leq j \leq m} u_j$ (i.e., move probability mass of the unvisited instances of U to u_i), and prune the object U from candidate set $\mathcal{C}$ at Line 12 if $\Upsilon(U^i) \leq \lambda$. Recall that λ is the k-th largest *rank scores* of the objects seen so far. Let U^0 denote an object with one instance u_0 where $s(u_0) = U_s^+$ and $p_{u_0} = 1.0$,i.e., an object constructed by pushing all instances of U to the lower corner of U_{mbr}. With similar rationale in Theorem 4.2, we have $\Upsilon(U^0) \geq \Upsilon(U)$. Consequently, at Line 18 of Algorithm 8, we can calculate $\Upsilon(U^0)$ based on U_s^+ without loading instances of U and remove U from candidate set $\mathcal{C}$ if $\Upsilon(U^0) \leq \lambda$. It is immediate that we can also remove any unvisited object $V \in \mathcal{C}$ (i.e., the object V with $V_s^+ < U_s^+$) in $\mathcal{C}$ from the top k candidates since we have $\Upsilon(V) \leq \Upsilon(V^0)$ and $\Upsilon(V^0) \leq \Upsilon(U^0)$.

4.2.5 Enhance the Performance with Statistics

Assume some statistics information (*mean* and *variance*) of the objects are available, we can further enhance the performance of the top k dominating query (Algorithm 8) by utilizing probabilistic inequalities. Specifically, given two objects U and V, we use $\Delta^-(V,U)$ ($\Delta^+(V,U)$) to denote the contribution of U towards the lower (upper) bound of the *dominance score* of V. In Section 4.2.2 we have $\Delta^-(V,U) = 0$ and $\Delta^+(V,U) = 1$ if V *partially dominates* U. This subsection shows that we can derive tighter lower and upper bounds for the *dominance scores* of the objects based on their MBRs and statistics information.

Motivation. As shown in Figure 4.4(a), given the MBR of an object U and a point p, we use rectangle A to denote the area of U_{mbr} on the left side of p (shaded

area). Similarly, in Figure 4.4(b), we use rectangle B to denote the area of U_{mbr} on the bottom of p (shaded area). Moreover, the rectangle R_p (the rectangle with thick line) represents the area of U_{mbr} which is dominated by the point p. Let $P(R)$ denote the probability mass of U within the rectangle R, then we should have $0 \leq \Delta^-(V,U) \leq P(R_p)$ if p corresponds to the upper corner of V_{mbr} (i.e., V^+_{mbr}). Since $P(R_p) \geq 1-(P(A)+P(B))$ in Figure 4.4(b), we may have $\Delta^-(V,U) = 1-(P(A)+P(B))$[3]. As we cannot have exact $P(A)$ $(P(B))$ value without accessing instances of U, this subsection shows that we can derive the upper bound of $P(A)$ $(P(B))$, denoted by $P^+(A)$ $(P^+(B))$ based on statistics information. Then we can come up with tight $\Delta^-(V,U)$ and $\Delta^+(V,U)$ values without loading instances of U. Specifically, we may have $\Delta^-(V,U) = 1-(P^+(A)+P^+(B))$ [4] (p is the upper corner of V_{mbr}) and $\Delta^+(V,U) = min(P^+(A), P^+(B))$ (p is the lower corner of V_{mbr}). With similar rationale, as shown in Figure 4.4(c)(d), we use rectangle A and B to denote the area of U_{mbr} on the right and top sides of p (shaded area) respectively, and the rectangle R_p (the rectangle with thick line) also represents the area of U_{mbr} which is dominated by the point p. If p corresponds to the lower corner of V_{mbr} (i.e., V^-_{mbr}), we should have $0 \leq \Delta^+(V,U) \leq P(R_p)$. Since $P(R_p) \leq P(A)$ and $P(R_p) \leq P(B)$, we may have $\Delta^+(V,U) = min(P(A), P(B))$. It is immediate that we get $\Delta^+(V,U) = min(P^+(A), P^+(B))$. With similar rationale, we may have $\Delta^+(V,U) = min(P^+(A), P^+(B))$ when p corresponds to the lower corner of V_{mbr} in Figure 4.4(d).

Example 4.4. *Suppose we have* $P^+(A) = 0.2$ *and* $P^+(B) = 0.3$ *in Figure 4.4. We have* $P(R_p) \geq 1-(P^+(A)+P^+(B)) = 0.5$. *Therefore, we can set* $\Delta^-(V,U) = 0.5$ *in Figure 4.4(b). Note that* $\Delta^-(V,U)$ *is set to* 0 *in Section 4.2.2 since* V *partially*

[3]We can set $\Delta^-(V,U)$ to any value τ with $0 \leq \tau \leq P(R_p)$ in this example.Obviously, the larger τ value the tighter lower bound we can achieve.

[4]We set $\Delta^-(V,U)$ to 0 if $P^+(A)+P^+(B) > 1$.

dominates U. Similarly, in Figure 4.4(d) we have $P(R_p) \leq min(P^+(A), P^+(B))$ and hence $\Delta^+(V, U)$ can be set to 0.2, which equals 1 in Section 4.2.2 since V partially dominates U.

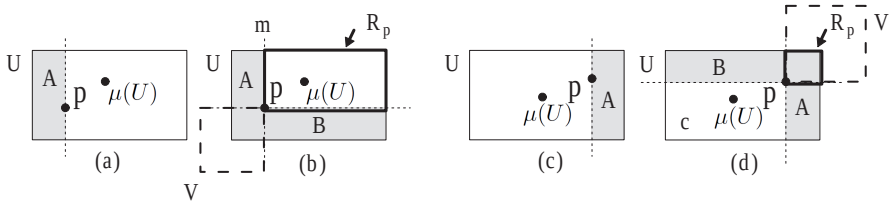


Figure 4.4: Statistic based Pruning

Definitions and Lemmas. We formally define two statistics information of an object U as follows.

Definition 4.3 (*mean* $\mu(U)$)**.** *We use $\mu(U)$ to denote the mean of an object U, where $\mu(U).D_i = \sum_{u \in U}(u.D_i \times p_u)$.*

Recall that $u.D_i$ is the i-th coordinate value of a point (instance) u.

Definition 4.4 (*variance* $\sigma^2(U)$)**.** *$\sigma^2(U)$ denotes the variance of an object U on each dimension; that is, $\sigma_i^2(U) = \sum_{u \in U}((u.D_i - \mu(U).D_i)^2 \times p_u)$.*

Given two values x and y, we use $\delta(x, y)$ to denote a function

$$\delta(x, y) = \begin{cases} \dfrac{1}{1 + \frac{x^2}{y^2}}, y \neq 0 \\ 1, x = 0 \text{ and } y = 0 \\ 0, x \neq 0 \text{ and } y = 0 \end{cases} \tag{4.9}$$

then *Cantelli's inequality* [Mee03] is defined as follows.

Lemma 4.1 (Cantelli's Inequality [Mee03])**.** *Suppose that t is a random variable in 1-dimensional space with mean $\mu(t)$ and variance $\sigma^2(t)$, $Prob(t - \mu(t) \geq a) \leq \delta(a, \sigma(t))$ for any $a \geq 0$, where $Prob(t - \mu(t) \geq a)$ denotes the probability of $t - \mu(t) \geq a$.*

Note that Lemma 4.1 extends the original Cantellis Inequality [Mee03] to cover the case when $\sigma = 0$ and/or $a = 0$. Then we can come up with another version of Cantelli's Inequality [LZZC11], which provides an upper-bound for $Prob(t \preceq b)$ when $b < \mu$.

Lemma 4.2. *Assume that* $0 < b < \mu(t)$. *Then,* $Prob(t \leq b) \leq \delta(\mu(t) - b, \sigma(t))$.

Proof. Let $t' = 2\mu(t) - t$. It can be immediately verified that $\sigma^2(t') = \sigma^2(t)$ and $\mu(t) = \mu(t')$. The theorem holds by applying Cantelli's Inequality on t'. □

Compute $\Delta^-(V, U)$. Given two objects U and V, the following theorem indicates that we can derive $\Delta^-(V, U)$ based on the statistics information.

Theorem 4.3. *Given two objects* U *and* V, *let* p *denote the upper corner of* V_{mbr}, *we have* $\Delta^-(V, U) = 1 - \sum_{1 \leq i \leq d}(\delta(\mu_i(U) - p.D_i, \sigma_i(U)))$ *if* $p \prec \mu(U)$. *Note that we set* $\delta(\mu_i(U) - p.D_i, \sigma_i(U))$ *to* 0 *if* $p.D_i < U^-_{mbr}.D_i$.

Proof. Let R_i denote the left side of the rectangle U_{mbr} divided by the point p on the i-th dimension (e.g., $R_1 = A$ and $R_2 = B$ in Figure 4.4(b)). We use $P(R_i)$ to record the probabilistic mass of the instances in U contained by R_i. Clearly, we have $P(R_i) = 0$ if $p.D_i < U^-_{mbr}.D_i$ since R_i corresponds to an empty rectangle. Otherwise, we have $P(R_i) \leq \delta(\mu_i(U) - p_i, \sigma_i(U))$ according to Theorem 4.2. Let R_p denote the rectangle whose lower (upper) corner is p (U^+_{mbr}), and $P(R_p)$ records the probabilistic mass of the instances in U contained by R_p. Then we have $P(R_p) \geq 1 - \sum_{1 \leq i \leq d}(\delta(\mu_i(U) - p.D_i, \sigma_i(U)))$, which implies that we can set $\Delta^-(V, U)$ to $1 - \sum_{1 \leq i \leq d}(\delta(\mu_i(U) - p.D_i, \sigma_i(U)))$ since all instances within R_p are dominated by p which is the upper corner of V_{mbr}. Therefore, the theorem holds. □

Compute $\Delta^+(V, U)$. The following theorem indicates that we can derive $\Delta^+(V, U)$ based on the statistics information.

Theorem 4.4. *Given two objects U and V, let p denote the lower corner of V_{mbr}, we have $\Delta^+(V,U) = min(\{\delta(p.D_i - \mu_i(U), \sigma_i(U))\})$ with $1 \leq i \leq d$ if $\mu(U) \prec p$ and $p \prec U^+_{mbr}$.*

We omit the details of the proof since it is similar to Theorem 4.3. The main difference is that Lemma 4.1 is employed in the proof instead of Lemma 4.2.

Achieve Better Dominance Score bounds. Let $\hat{V}_s^-$ ($\hat{V}_s^+$) denote the new lower (upper) bound for the *dominance score* of the object V, and $\mathcal{S}$ is the set of objects which are *partially dominated* by V, we have

$$\hat{V}_s^- = V_s^- + \sum\nolimits_{U \in \mathcal{S}} \Delta^-(V,U) \tag{4.10}$$

and

$$\hat{V}_s^+ = V_s^- + \sum\nolimits_{U \in \mathcal{S}} \Delta^+(V,U) \tag{4.11}$$

Recall that we have $\Delta^-(V,U) = 0$ and $\Delta^+(V,U) = 1$ in Section 4.2.2 where only MBRs of the objects are used.

Suppose the statistics information of the objects are kept with the data entries of the objects in the *global* R-tree, we can use the new lower and upper bounds of *dominance scores* in Algorithm 8. Our empirical study shows that although the statistics information slightly increase the index size, the gain is significant since the tighter *dominance scores* bounds lead to smaller candidate size, and hence reduce the CPU and I/O costs.

4.3 Experiment

In this section, we present results of a comprehensive performance study to evaluate the efficiency and scalability of the proposed techniques in the chapter. As there

is no existing work on top k dominating query on uncertain data following the *parameterized ranking* semantics, we only evaluate the techniques proposed in the chapter. Following algorithms are implemented for performance evaluation.

- **NAIVE:** The *straightforward solution* is introduced in the Section 4.2. Specifically, we first compute the *dominance score* of each instance by issuing range queries against the global R-tree and local R-trees, then Algorithm 7 is employed to compute top k dominating objects.
- **NAIVES:** Algorithm 8 proposed in Section 4.2 where only the spatial based pruning technique (Section 4.2.3) is employed.
- **BAS:** Algorithm 8 proposed in Section 4.2 where spatial based pruning technique and spatial join based *dominance score* computation algorithms (Section 4.2.2) are employed. It is employed as the baseline algorithm in our empirical study.
- **TKDOM: BAS** Algorithm which also applies the *rank score* based pruning technique (Section 4.2.4).
- **TKDOM*: TKDOM** Algorithm which also applies statistics based techniques (Section 4.2.5).

We employ a specific *parameterized ranking linear* function $PFR^e(\alpha)$ to rank the objects. Like the setting in [LSD11], we use $PFR^e(\alpha = 0.95)$ in the experiments.

Datasets We evaluate our techniques on both synthetic and real datasets. Synthetic datasets are generated by using the methodologies in [BKS01] regarding the following parameters. Dimensionality d varies from 2 to 5 with default value 3. Data domain in each dimension is [0, 10000]. The number n of objects in each

dataset varies from 10K to 50K with default value 10K. The number m of instances per object varies from 50 to 300 with the default value 100. The value k varies from 10 to 50 with default value 20. The edge length h of object MBRs varies from 100 to 600 with default value 200. Centers of objects (objects' MBRs) follow either Equally(E), Correlated(C) or Anti-correlated(A) distribution where default is A distribution.The instances of an uncertain object follow popular distributions Normal(N) and Uniform(U) where N distribution is default. And two real datasets, Forest CoverType dataset (COV)[5] and Household (HOU)[6], are employed to represent the centers of the uncertain objects. In COV, we select the horizontal and vertical distances of each observation point to the Hydrology as well as the elevation of the point. In HOU, each record represents the percentage of an American family's annual income spent on 3 types of expenditures (e.g., gas, etc.). We choose $20,000$ objects in COV and HOU respectively. For each object, we generate the instances according to the default setting above. Then with the default setting, the total number of instances in synthetic and real datasets are 1 millions and 2 millions respectively. Furthermore, we used an LRU memory buffer whose default size is set to 10% of the data size.

All algorithms are implemented in standard C++ and compiled with GNU GCC. Experiments are run on a PC with Intel Xeon 2.40GHz dual CPU and 4G memory under Debian Linux. The disk page size is fixed to $4,096$ bytes. In the chapter, we evaluate the I/O performance of the algorithms by measuring the number of uncertain objects explored. i.e., uncertain objects whose aggregate R-tree are loaded in main memory. Query response time is recorded to evaluate the efficiency of the algorithms, which contains the CPU time and the I/O latency. Table 4.2 lists all parameters which may have impacts on our performance study,

[5] `http://archive.ics.uci.edu/ml/datasets.html`

[6] `http://www.ipums.org`

where the default values are in **bold** font. In our experiments, all parameters use default values unless otherwise specified.

Notation	Definition (Default Values)
dimensionality d	2, **3**, 4, 5,
number of objects n	**10K**, 20K, 30K, 40K, 50K
number of instances m	50, **100**, 200, 300
edge length h	100, **200**, 400, 600
top k	10, **20**, 30, 40, 50
object location	equally, correlated, **anti-correlated**
instance location	**normal**, uniform

Table 4.2: Parameter settings

Performance Evaluation

In the first experiment, we vary the value of k and evaluate the performance of the five algorithms. Figure 4.5 illustrates the response time and the number of objects accessed for algorithms NAIVE, NAIVES, BAS, TKDOM and TKDOM* against the default synthetic dataset where k varies from 10 to 50.As expected, all techniques proposed in Section 4.2 are effective since the performance of them degrades slowly against the growth of k. When the k grows, we get a larger candidate set, so it is require more time and access more objects to get the result. There is a clear gap between any two consecutive algorithms. Note that the NAVIE and NAIVES algorithms are much slower than the other algorithms, and the I/O costs of them are also much higher than the others. Then for better report on the performance of the algorithms, we exclude the NAIVE and NAIVES algorithms in the following experiments since they have been significantly outperformed by BAS, TKDOM and TKDOM* algorithms.

Impact of Data Distribution. We evaluate the performance of BAS, TKDOM and TKDOM* against datasets C_N, C_U, E_N, E_U, A_N, A_U, COV and HOU in

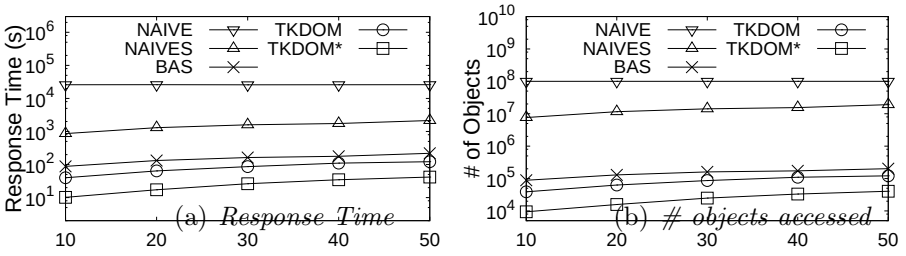


Figure 4.5: Diff. top k

Figure 4.6, where C_N denotes the 3-dimensional synthetic data whose centers and instances follow the Correlated and Normal distributions respectively, and similar definitions go to C_U, E_N, E_U, A_N and A_U. It is observed that the distribution of the instances (N and U) does not noticeably affect performance of the algorithms, so we only perform tests on normal distribution in the following experiments. On the other side, all algorithms are very sensitive to the distribution of the object centers. This is because of the nature of dominating query. It is easy to distinguish and sort the *dominance scores* of objects to get a small size of candidate set under the correlated distribution.Whereas anti-correlated distribution leads to more computation time because each object only fully dominate a limited number of other objects, so we use anti-correlated distribution as a default setting for locations. As expected, TKDOM* significantly outperforms other algorithms under all data distributions.

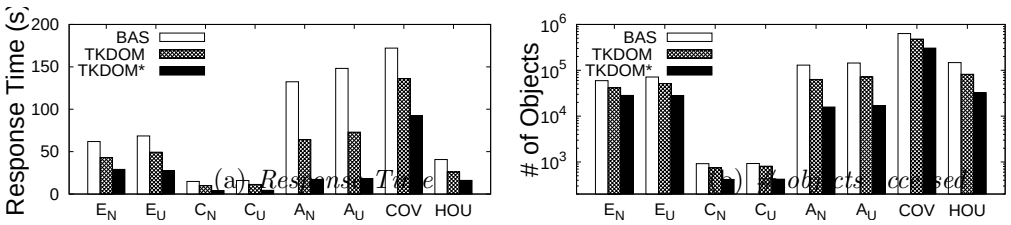


Figure 4.6: Impact of Data Distributions

Impact of dimensionality (d). Figure 4.7 investigates the impact of the dimensionality against the algorithms where d varies from 2 to 5. It is shown that all methods are sensitive to the growth of dimensionality. The response time and the number of I/O accessed decrease when d arises. This is because the average number of objects *dominated* or *partially dominated* by each object decreases against the growth of the dimensionality. That means with the increase of dimensionality d, the average area of MBRs gets smaller compared to the whole data space; consequently, the power of the pruning rules becomes more significant. Note that the edge lengths of the objects remains unchanged when the dimensionality grows in the experiments.

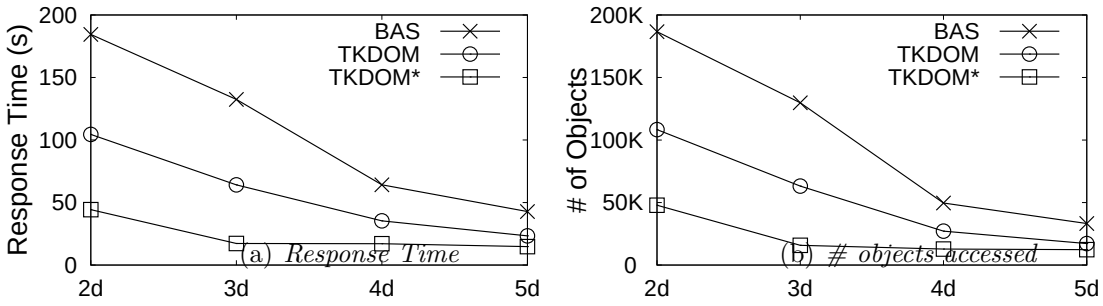


Figure 4.7: Diff. dimensionality

Impact of the edge length (h). We investigate the performance of the algorithms as a function of the edge length which varies from 100 to 600. With large h values, the MBRs of objects are more likely to overlap with each other and hence the pruning power is impaired. Figure 4.8 shows that the scalability of TKDOM* is better than that of the other two algorithms regarding the growth of h.

Impact of the number of objects (n). In Figure 4.9, we evaluate the impact of the number of objects against the three algorithms, where the number of objects varies from 10K to 50K. With a larger number of objects, more objects are involved in the computation, thus incurring higher computation cost. Figure 4.9 show that

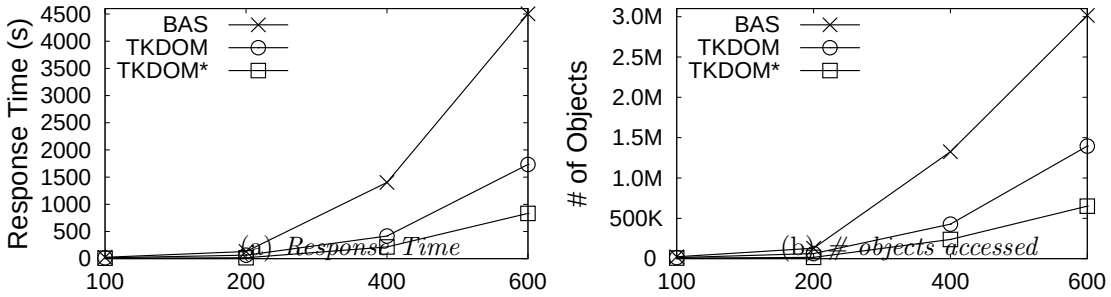


Figure 4.8: Diff. edge length

the response time and the number of objects accessed of TKDOM* grow slowly, yet the performance of the BAS and TKDOM drops more significantly with the growth of n. It is clear that TKDOM* has the best scalability among three algorithms.

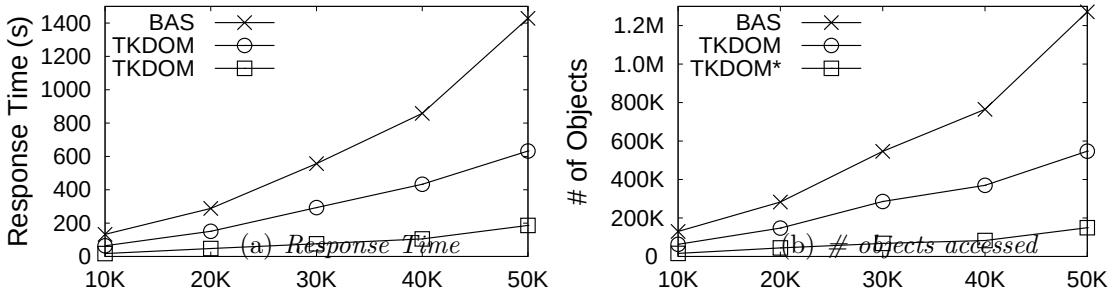


Figure 4.9: Diff. # objects

Impact of the number of instances (m). Figure 4.10 evaluates the impact of the number of instances on the algorithms where m grows from 50 to 300. The number of objects accessed grows slowly against each algorithm when the number of instances increase, which show our global pruning techniques are effective and efficient. With growth of m, more instances are involved in the rank computation, thus the response time increases. Figure 4.10 shows that TKDOM* has the best scalability against m, followed by TKDOM and BAS.

Summary. As a short summary, our comprehensive performance study shows that

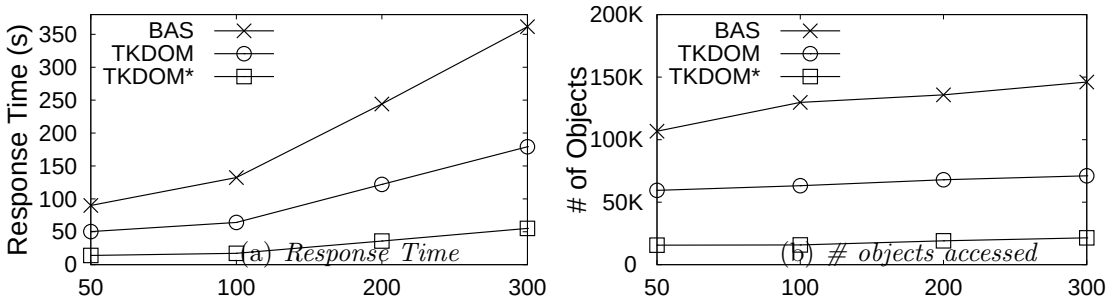


Figure 4.10: Diff. # instances

the effectiveness and efficiency of the techniques proposed in the chapter, including spatial join based *dominance score* computation algorithm (Section 4.2.2), spatial pruning (Section 4.2.3), *rank score* based pruning (Section 4.2.4), and statistics based computation techniques (Section 4.2.5). As expected, the TKDOM* algorithm, which includes all techniques proposed in the chapter, always outperforms other algorithms under all experiment settings.

4.4 Conclusion

We investigate the problem of identifying top k dominating objects over uncertain data, which is important in the multi-criteria decision analysis when users cannot explicitly provide a proper scoring function. Based on the state-of-the-art top k semantics on uncertain data, we formally define a new model for the top k dominating query on multi-dimensional uncertain data. By utilizing the popular R-tree indexing techniques as well as spatial based and rank score based pruning techniques, we develop an effective and efficient algorithm following the *filtering and verification* paradigm. We further improve the performance of the algorithm based on some simple statistics information of the objects. Our experiments convincingly demonstrate the effectiveness and efficiency of our techniques.

Chapter 5

Range Search on Uncertain Trajectories

A straightforward approach for the problem of range search on uncertain trajectories is to calculate the *appearance probability* of each moving object o at each time within the query time interval, and count the number of times in which o appears within the search region with probability at least θ. In this chapter, we follow the *filtering-and-refinement* paradigm to significantly reduce the number of candidate trajectories (i.e., moving objects) by exploiting effective filtering techniques. In particular, for any two subsequent observations of a moving object o at times t_i and t_j, denoted by $o(t_i)$ and $o(t_j)$ respectively, we aim to build a summary of the uncertain location distribution of the object. Thus, lower and upper bounds of its appearance probability can be easily derived for any time $t \in (t_i, t_j)$ when a range query is issued. Novel sub-diamonds based filtering technique is proposed in [EKM$^+$12a] to effectively support range search on uncertain trajectories. However, we observe that its performance is unsatisfactory in our empirical study, because each sub-diamond aims at bounding the appearance probability of the moving object o regarding **all**

times between two subsequent observation times t_i and t_j. This motivates us to develop new filtering techniques so that the summary is constructed for a set of time intervals instead of the whole time interval (t_i, t_j) (i.e., partition along the temporal dimension). Specifically, we first introduce a simple filtering technique based on some pre-computed statistics information. Then we further enhance the filtering power by developing partition based approach to approximate the location distribution of a moving object using a set of buckets, which are generated by spatial and temporal partitions. We discuss how to effectively build the partition based summaries following some important observations.

This chapter is organized as follows. Section 5.1 formally defines the problem of range search on uncertain trajectories and introduce some preliminary work. Section 5.2 introduces a general framework for range search following the filtering and refinement paradigm. Section 5.3 and Section 5.4 propose the statistics based and partition based filtering techniques respectively. Experimental results are reported in Section 5.5, and Section 5.6 concludes the chapter.

5.1 Background

In this section, we first formally define the problem of range search on uncertain trajectories. Then we introduce the preliminary work on uncertain trajectories. Table 5.1 summarizes the notations frequently used throughout this chapter.

5.1.1 Problem Definition

Following the common assumptions of the existing works (e.g., [EKM$^+$12b, EKM$^+$12a, NZE$^+$13, XGC$^+$13]) which capture the uncertainty of trajectories with Markov Chain model, we assume the space and time are in discrete domain. The

space $\mathcal{S}$ consists of n possible states (locations) $\{s_1, \ldots, s_n\}$ in 2-dimensional space. For a state s, $s.D_i$ denotes the coordinate value of s on i-th dimension. We use $\mathcal{T}$ to denote time domain $\{t_1, \ldots, t_m\}$. Consequently, in this chapter, the trajectory of a moving object o is represented by a set of m' ($m' \leq m$) tuples $\{t_k, o(t_k)\}$, where $o(t)$ represents the state of o at time t. For a certain trajectory, $o(t)$ is an unique state $s \in \mathcal{S}$. However, it corresponds to a probability distribution when the location of the object is derived from probabilistic models. Following is a formal definition of the uncertain location for an object o at time t.

Notation	**Definition**
o ($\mathcal{O}$)	a moving object (a set of moving objects)
$\mathcal{S}$ ($\mathcal{T}$)	discrete space (time) domain
$o(t)$	location (state) of an object o at time t
q	spatio-temporal search region
R	a spatial search region
$[q.s, q.e]$	query time interval
θ	probabilistic threshold
η	duration threshold
$P(o(t), s)$	probability that $o(t)$ is located at sate s
$g(o, t_i, t_j), g$	a segment of an object o with two subsequent observations $o(t_i)$ and $o(t_j)$
$\mathcal{T}(g)$	time interval $[q.s, q.e] \cap [t_i, t_j)$
$\Delta_t(q)(\Delta_t(g))$	duration of a query (segment)
$P(o(t), R)$	probability that $o(t)$ falling in the region R
$d(o, q, \theta), d(o)$	duration (i.e., number of times) that object o satisfies the search region $q.R$
$d^-(o)$ $(d^+(o))$	lower (upper) bound of $d(o)$
$d(g)$	number of satisfied times of the object o in segment g
$d^-(g)$ $(d^+(g))$	lower (upper) bound of d(g)
$\mathcal{S}(g)$	partition based summary of segment g
$g.mbr$	minimal bounding rectangle of segment g

Table 5.1: The summary of notations.

Definition 5.1 (Uncertain Location)**.** *Let $P(o(t), s)$ denote the probability that an object o appears on state (location) s at time t. The uncertain location of an object o*

at time t, denoted by $o(t)$, consists of n' tuples $\{s_i, P(o(t), s_i)\}$ with $P(o(t), s_i) > 0$, where $\sum_{i=1}^{n'} P(o(t), s_i) = 1$.

Consequently, an **uncertain trajectory** is a trajectory whose location might be uncertain at each point of time. In particular, we assume the location (state) of an object is certain when it is observed (reported), while locations (states) of an object between two subsequent observation times are derived based on the Markov Chain model [EKM+12a] which is introduced in Section 2.4.2. Therefore, the uncertain trajectory of an object o consists of a set of **segments** $\{g(o, t_i, t_j)\}$ where t_i and t_j corresponds to two subsequent observations. Each segment g records the location distribution of the object o from time t_i (inclusive) to t_j (exclusive). We use $\Delta_t(g)$ to denote the duration of the segment (a.k.a. observation interval size) where $\Delta_t(g) = t_j - t_i$.

Given a region R, we use $s \in R$ to denote that the location (state) s is within the region R. Then we define the **appearance probability** of o regarding R at time t, denoted by $P(o(t), R)$, to measure the likelihood of the object o falling in the region R at time t.

$$P(o(t), R) = \sum_{s \in R \text{ and } s \in \mathcal{S}} P(o(t), s) \tag{5.1}$$

Definition 5.2 (Spatio-temporal search region (q))**.** *A spatio-temporal search region q consists of three components $\langle R,\ s,\ e \rangle$, where $q.R$ represents the spatial search region, and $q.s$($q.e$) denotes the start (end) time of the query time interval. We use $\Delta_t(q)$ denotes the duration of the search region (i.e., $\Delta_t(q) = q.e - q.s + 1$).*

We say an object satisfies the spatio search region $q.R$ with probability at least θ at time t if $P(o(t), q.R) \geq \theta$ for $t \in [q.s, q.e]$. We use $d(o, q, \theta)$ to denote the duration (i.e., the number of times) that a moving object o satisfies $q.R$ with probability

at least θ. For presentation simplicity, we use $d(o)$ to represent $d(o, q, \theta)$ whenever there is no ambiguity.

Finally we have the formal definition of the spatio-temporal range search on uncertain trajectories.

Problem Statement. In this chapter, we investigate the problem of spatio-temporal range search over uncertain trajectories. Particularly, given a spatio-temporal search region q, a probabilistic threshold θ ($0 < \theta \leq 1$), a duration threshold η ($1 \leq \eta \leq \Delta_t(q)$), and uncertain trajectories of a set $\mathcal{O}$ of moving objects, we aim to identify objects $\{o|\ d(o, q, \theta) \geq \eta\}$ with $o \in \mathcal{O}$; that is, find objects which consistently (at least η times) appear within the spatial search region with probability at least θ.

Range queries on uncertain trajectories with *EXISTS* and *ALL* semantics in [EKM+12a, EKM+12b] are special cases of the problem studied in this chapter, which correspond to the range search with $\eta = 1$ and $\eta = \Delta_t(q)$, respectively.

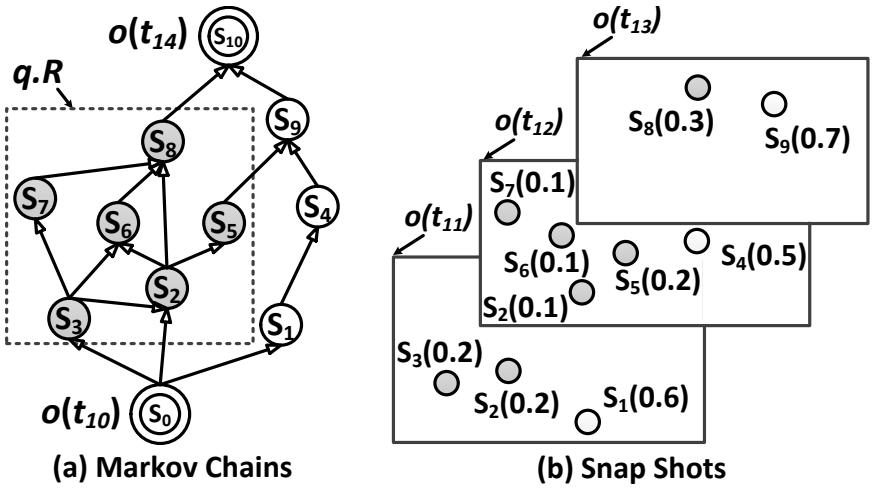


Figure 5.1: Range Search over Uncertain Trajectories

Example 5.1. *In Figure 5.1(a), $o(t_{10})$ and $o(t_{14})$ are two subsequent observations of object o; a spatio-temporal search region q is given as $\langle R, t_{11}, t_{13}\rangle$ with probabilistic threshold θ and duration threshold η. The snap shots of o are depicted*

in Figure 5.1(b) for t_{11}, t_{12} and t_{13}, where the appearance probability for each state is marked. Note that shaded states are contained by $q.R$. Then we have $P(o(t_{11}), q.R) = 0.4$, $P(o(t_{12}), q.R) = 0.5$ and $P(o(t_{13}), q.R) = 0.3$. Given $\eta = 2$, if $\theta \leq 0.4$, o meets the query constraints, otherwise o is not an answer.

Thereafter of this chapter, the "spatio-temporal range search" is abbreviated to "range search", and "spatio-temporal search region" are abbreviated to "search region" for presentation simplicity. We might use "location" and "state" interchangeably for better understanding of this chapter.

5.1.2 Preliminary

Recent years have witnessed the increasing amount of research on uncertain data modeling and query processing due to their importance in many applications. In this subsection, we briefly introduce two categories of work closely related to the problem studied in this chapter.

Capture Uncertainty by Markov Chains

In [EKM+12b, EKM+12a], an uncertain trajectory is modeled as a realization of a stochastic process [KT75]. Markov Chains can model a discrete spatio-temporal (state-time) space with the assumption that $o(t+1)$ only depends on $o(t)$.

Definition 5.3 (Markov Chain model)**.** *Given a stochastic process $o(t)$ with $t \in \mathcal{T}$ and a state $s \in \mathcal{S}$, the stochastic process is called Markov Chain iff $P(o(t+1) = s_j | o(0) = s_0, o(1) = s_1, \ldots, o(t) = s_i) = P(o(t+1) = s_j | o(t) = s_i)$.*

For an object o moving on the space $\mathcal{S}$, we set $P_{i,j}(o) = P(o(t+1) = s_j | o(t) = s_i)$, where $P_{i,j}(o)$ represents the probability of object o moving from state s_i to s_j when the time changes from anytime time $t \in \mathcal{T}$ to its successive time $t+1$. We can

store all $P_{i,j}(o)$ in a $n \times n$ matrix $M(o)$ to represent the transition probability of object o from state s_i to s_j at any time t, where the matrix $M(o)$ is called transition matrix. Then we have $o(t+1) = o(t) \times M(o)$. Recall that $o(t)$ is the distribution vector of an object o at time t where $\sum_{s \in \mathcal{S}} P(o(t), s) = 1$. Similarly, if $M(o)^T$ is defined as a transposed Markov Chain matrix, we have $o(t) = o(t+1) \times M(o)^T$.

Given two subsequent observations $o(t_i)$ and $o(t_j)$, efficient algorithm is proposed in [EKM+12b] to derive the location distribution $o(t)$ for $t \in (t_i, t_j)$ based on $M(o)$ and $M(o)^T$. Same as [EKM+12b, EKM+12a], we assume objects share the same Markov Chain matrix which can be learned by domain experts in various applications.

In an uncertain object trajectory of an object o, we have several consequent observation timestamps. The position of the object on each timestamp is certain, whereas the positions between two continuous timestamps are uncertain, which can be modeled by the two transition matrixes. For example, if we observe the object 0 on s_i when t_1 and then s_j when t_5, the positions of o on t_2, t_3, t_4 are unknown. Let $P(o, t_1) = (0|s_1, \ldots, 1|s_i, \ldots, 0|s_n)$, then the $P_f(o, t_2) = P(o, t_1) \times M(o)$, $P_f(o, t_3) = P(o, t_1) \times M(o)^2$, $P_f(o, t_4) = P(o, t_1) \times M(o)^3$. Similarly, Let $P(o, t_5) = (0|s_1, \ldots, 1|s_j, \ldots, 0|s_n)$, then the $P_b(o, t_4) = P(o, t_5) \times M(o)^T$, $P_b(o, t_3) = P(o, t_5) \times (M(o)^T)^2$, $P_b(o, t_4) = P(o, t_5) \times (M(o)^T)^3$. Note that P_f and P_b mean the result derived from $M(o)$ and $M(o)^T$ respectively. After combining the P_f and P_b, we can get the distributions of the object on t_2, t_3, t_4, and then the possible worlds are gotten.

Sub-diamonds based Filtering

The sub-diamonds based filtering technique developed in [EKM+12a] can significantly reduce the computational cost compared with the diamond based technique.

However, as we need enforce that the object o appears within the sub-diamond $\Diamond$ with probability at least $P(\Diamond)$ w.r.t **all** $t \in (t_i, t_j)$, this may lead to poor filtering performance. In our empirical study, we observe that the corresponding probability (i.e.,$P(\Diamond)$) of the sub-diamonds $\{\Diamond\}$ might be rather small, especially when the timespan between t_i and t_j is long. Moreover, sub-diamonds are calculated based on the projected values for each individual dimension separately. As reported in [ZLZ+10b], this may lose the spatial correlation of the object location distribution, and hence deteriorates the filtering performance. These problems cannot be addressed by simply increasing the number of sub-diamonds.

5.2 Framework

A straightforward implementation of range search on uncertain trajectories is to calculate $d(o, q, \theta)$ for each individual object $o \in \mathcal{O}$ through Markov Chains based computation technique [EKM+12b]. However, as shown in [EKM+12a], this is cost-prohibitive since the *refinement* cost is rather expensive. Therefore, it is desirable to develop effective and efficient *filtering* techniques to **prune** or **validate** objects such that the number of objects involving refinement can be significantly reduced. In particular, suppose we can derive the upper and lower bounds for the duration of an object o regarding the range search, denoted by $d^+(o)$ and $d^-(o)$ respectively. Then an object can be safely pruned if $d^+(o) < \eta$ or validated if $d^-(o) \geq \eta$. Moreover, for each individual time t, we can also derive lower and upper bounds of the appearance probability, denoted by $P^-(o(t), q.R)$ and $P^+(o(t), q.R)$, so we can avoid the computation of $P(o(t), q.R)$ if $P^-(o(t), q.R) \geq \theta$ (*validate*) or $P^+(o(t), q.R) < \theta$ (*prune*).

In this chapter, we develop efficient algorithms to support range search on

uncertain trajectories following filtering and refinement paradigm. In the sequel, we first introduce a simple minimal bounding rectangle based filtering technique, then present a general framework for range search on uncertain trajectories.

5.2.1 Minimal Bounding Rectangle Based Filtering

Given two subsequent observations $o(t_i)$ and $o(t_j)$, we may easily come up with a minimal bounding rectangle (MBR) for each segment g, denoted by $g.mbr$, which encloses all possible locations of o during time t_i and t_j if the maximal speed is pre-given. Together with the time dimension, each segment g can be enclosed by a 3-dimensional minimal bounding rectangle (MBR), denoted by $g.mbr$. Clearly, a spatio-temporal search region q is a cube in 3-dimensional space. In this chapter, we define three relations between q and $g.mbr$. We say a query q *does not overlap* a segment g if q and $g.mbr$ does not overlap w.r.t spatial or temporal aspects; that is, $q.R \cap g.mbr = \emptyset$, $q.s \geq t_j$ or $q.e < t_i$. Otherwise, we say q **overlaps** g. Particularly, we say q **contains** g if g is contained by q on both spatial and temporal aspects, i.e., $g.mbr \subset q.R$, $q.s \leq t_i$, and $q.e \geq t_j$. Let $d(g)$ denote the contribution of the segment g to $d(o)$ where $0 \leq d(g) \leq \Delta_t(g)$. It is immediate that we have $d(g) = \Delta_t(g)$ if q contains g, and $d(g) = 0$ if q does not overlap g.

5.2.2 Segments Summaries Tree (SS-Tree)

MBR based filtering technique is simple and intuitive, and its filtering capability is rather limited. In Section 5.3 and Section 5.4, we introduce advanced filtering techniques based on statistics information and spatio-temporal partitions of the segment respectively where summaries of the segments are pre-computed to facilitate the filtering process.

In this chapter, we assume a summary of the segment is constructed for each in-

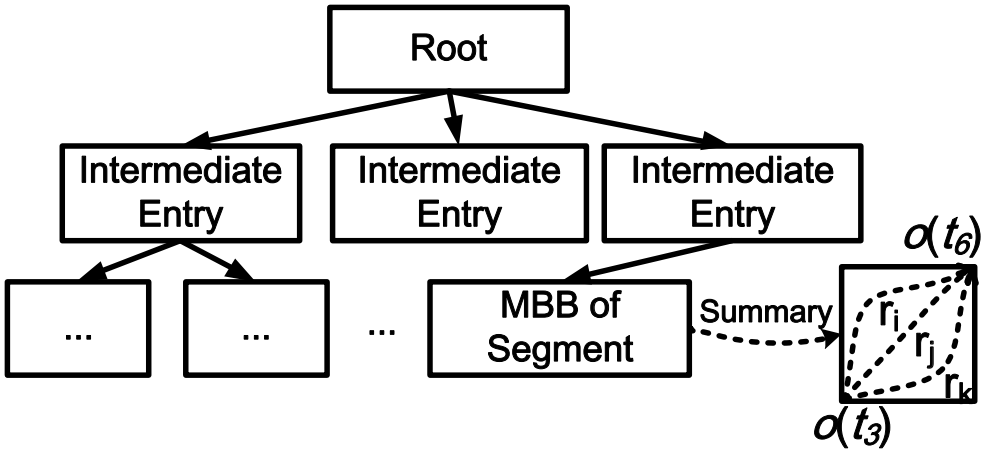


Figure 5.2: Segments Summaries Tree

dividual segment w.r.t the filtering technique used (e.g., sub-diamonds based filtering, statistics based filtering and partition based filtering). As shown in Figure 5.2, MBRs of the segments are organized by a hierarchical spatial index structure (e.g., R-tree [Gut84]). For each segment entry, its corresponding summary is maintained to enhance the filtering performance.

With the same rationale to MBR based filtering in Section 5.2.1, we can easily come up with three relations between q and an intermediate entry E (i.e., segments enclosed by E). Then an intermediate entry E can be *pruned* or *validated* without further exploring its child entries.

5.2.3 A General Framework

Assuming the summaries of the segments are organized by an SS-tree, we present a general framework for the range search on uncertain trajectories following the filtering and refinement paradigm, and details are illustrated in Algorithm 10.

In particular, we traverse the entries in a branch and bound fashion. A FIFO queue, denoted by $\mathcal{Q}$, is used to maintain the entries to be visited. In Line 5-13, entries are processed according to their relationships with search region q. Clearly,

Algorithm 10: ***Range Search***($\mathcal{O}_T$, q, θ, η)

Input : $\mathcal{O}_T$: Uncertain trajectories of a set $\mathcal{O}$ of objects organized by SS-Tree T, q : range search, θ : probabilistic threshold, η : duration threshold

Output: objects $\{o\}$ with $d(o, q, \theta) \geq \eta$

$\mathcal{C} := \emptyset$; $\mathcal{F} := \emptyset$; $\mathcal{R} := \emptyset$;
$\mathcal{Q} \leftarrow$ push root of $\mathcal{O}_T$;
while $\mathcal{Q} \neq \emptyset$ **do**
    $E :=$ element popped from $\mathcal{Q}$;
    if *E overlaps q* **then**
        if *E is contained by q* **then**
            for *each segment $g \in E$ of object o* **do**
                $d^-(o) := d^-(o) + \Delta_t(g)$;
                $\mathcal{R} := \mathcal{R} \cup o$ **If** $d^-(o) \geq \eta$; `// validate`
        else if *E is an intermediate entry* **then**
            Push child entries of E into $\mathcal{Q}$;
        else
            $\mathcal{F} := \mathcal{F} \cup$ corresponding segment g;
$d^+(o) = d^-(o)$ for all objects with segments in $\mathcal{F}$;
for *each segment g of object o in $\mathcal{F}$* **do**
    $d^+(o) := d^+(o) + \Delta_t(g)$;
$\mathcal{C} \leftarrow$ objects $\{o\}$ if $d^+(o) \geq \eta$ and $o \notin \mathcal{R}$; `// prune`
for *each candidate object $o \in \mathcal{C}$* **do**
    for *each candidate segment g of o in $\mathcal{F}$* **do**
        Derive $d^-(g)$ and $d^+(g)$ from summary associated with g;
        $\mathcal{F} := \mathcal{F} \setminus g$ **If** $d^-(g) = \Delta_t(g)$ or $d^+(g) = 0$;
        $d^-(o) := d^-(o) + d^-(g)$;
        $d^+(o) := d^+(o) - \Delta_t(g) + d^+(g)$;
    if $d^-(o) \geq \eta$ **then** `// validate`
        $\mathcal{R} := \mathcal{R} \cup o$; $\mathcal{C} := \mathcal{C} \setminus o$;
    else if $d^+(o) < \eta$ **then** `// prune`
        $\mathcal{C} := \mathcal{C} \setminus o$;
for *each object o in $\mathcal{C}$* **do** `// refinement`
    $\mathcal{R} := \mathcal{R} \cup o$ **If** o is verified;
return $\mathcal{R}$;

we do not need to further explore an entry E if it does not overlap q (Line 5), because none of the segments enclosed by E contribute to the final results. On the other hand, if E is *contained* by q, we can immediately *validate* the segments $\{g\}$ enclosed by E; that is, we increase $d^-(o)$ by $\Delta_t(g)$ (Line 8) where g is a segment of the object o. Line 9 validates o if we have $d^-(o) \geq \eta$ where $\mathcal{R}$ is used to keep query results. Otherwise, i.e., E *overlaps* q but is not contained by q, Line 10-13 further explore an entry by expanding its child entries or put its corresponding segment into the set $\mathcal{F}$ which will be further processed by advanced filtering technique at Line 20.

We update $d^+(o)$ by accumulating the durations of its validated segments (Line 14) and unexplored segments, i.e., segments in $\mathcal{F}$ (Line 16). Line 17 retrieves candidate objects which are not validated but have a promising upper bound (i.e., $d^-(o) < \eta$ and $d^+(o) \geq \eta$). Then Line 18-27 further refine the candidate set by exploiting the advanced filtering techniques, in which the summary of segment may derive tighter lower and upper bounds for $d(g)$. Note that Line 21 removes a segment g from candidate segments $\mathcal{F}$ if we have $d^-(g) = \Delta_t(g)$ (i.e., o is qualified at all times $t \in [t_i, t_j)$) or $d^+(g) = 0$ (i.e., o is not qualified at any time $t \in [t_i, t_j)$).

Finally, Line 28-29 refine the remaining candidate objects by exactly computing $d(o, q, \theta)$ for each candidate object o. Note that we only need to compute $d(g)$ for candidate segments $\{g\}$ in $\mathcal{F}$.

As shown in our empirical study, the dominant cost of Algorithm 10 is the refinement cost at Line 29, since it is time consuming to calculate appearance probabilities of objects at different times. This motivates us to develop effective and efficient filtering techniques to significantly reduce the number of survived candidates with reasonable space overhead. In Section 5.3 and Section 5.4, we present advanced filtering techniques based on statistics information and spatio-

temporal partitions respectively.

5.3 Statistics Based Approach

In this section, we present the statistics based filtering technique. Section 5.3.1 introduces the motivation of the technique. Section 5.3.2 proposes the detailed pruning and validation rules. Performance analysis is conducted in Section 5.3.3.

5.3.1 Motivation

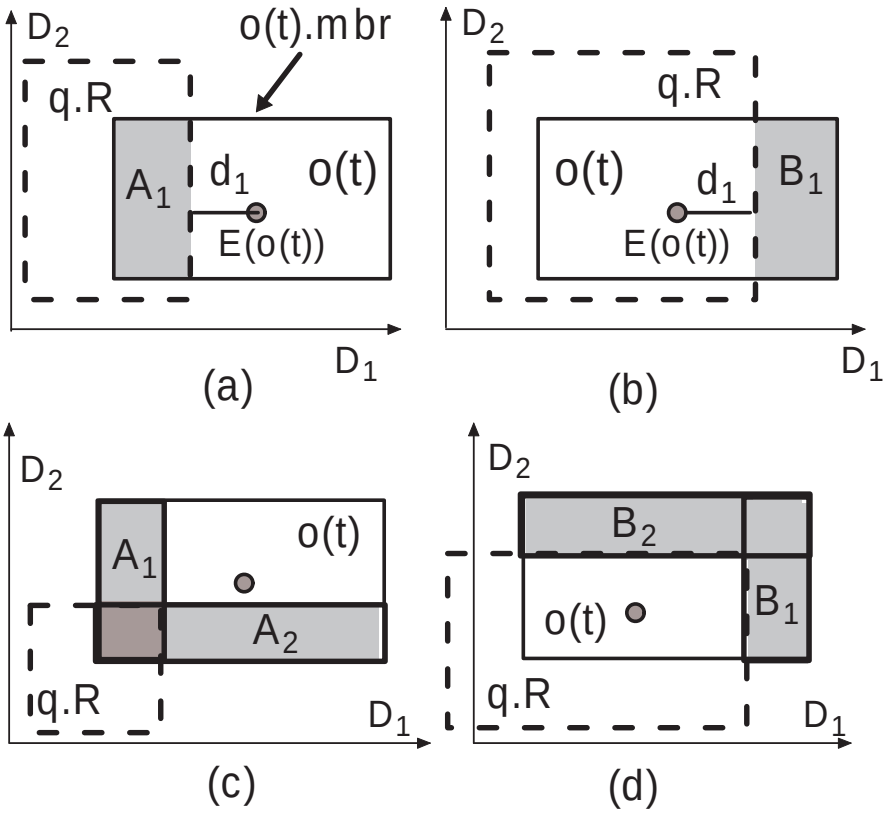


Figure 5.3: Motivation of Statistics based Filtering

In this section, we develop the statistics information based filtering technique. In a nutshell, for each segment $g(o, t_i, t_j)$ we use some simple statistics to capture the

uncertain location distribution of the object for each time $t \in (t_i, t_j)$. Then we can derive lower and upper bounds of the appearance probability of o at time t, denoted by $P^-(o(t), q.R)$ and $P^+(o(t), q.R)$ respectively, to *prune* or *validate* the time t. We say a time t is pruned (validated) if $P^+(o(t), q.R) < \theta$ ($P^-(o(t), q.R) \geq \theta$). Furthermore, we can merge the statistics of a set of consecutive times to reduce the summary size.

As shown in Figure 5.3, suppose $o(t)$ is bounded by a minimal bounding rectangle, denoted by $o(t).mbr$. We use $P(A_i)$ $(i = 1, 2)$ to denote the probability mass of the states (locations) which are *contained* by $q.R$ along the i-th dimension D_i, and $P(B_i)$ records the probability mass of the states which are *not contained* by $q.R$ along the i-th dimension. In consequence, we have $P(o(t), q.R) = P(A_1)$ in Figure 5.3(a), since $q.R$ contains $o(t)$ along the dimension D_2. Suppose we can derive an upper bound of $P(A_1)$, denoted by $P^+(A_1)$. Then we can safely prune the time t, if $P^+(A_1) < \theta$. Similarly, t is validated in Figure 5.3(b) if $1 - P^+(B_1) \geq \theta$, where $P^+(B_1)$ is the upper bound of $P(B_1)$. This observation can be easily extended to the case where the search region overlaps $o(t).mbr$ on both dimensions, e.g., Figure 5.3(c) and Figure 5.3(d). In Figure 5.3(c), we have $P(o(t), q.R) \leq min(P(A_1), P(A_2)) \leq min(P^+(A_1), P^+(A_2))$, and we can prune t if $min(P^+(A_1), P^+(A_2)) < \theta$. With the similar rationale, t is validated if $1 - (P^+(B_1) + P^+(B_2)) \geq \theta$.

Example 5.2. *Suppose we have* $P^+(A_1) = 0.1$, $P^+(B_1) = 0.1$, $P^+(A_2) = 0.2$, *and* $P^+(B_2) = 0.2$ *in Figure 5.3. Then we have* $P(o(t), q.R) \leq 0.1$ *in Figure 5.3(a),* $P(o(t), q.R) \geq 0.9$ *in Figure 5.3(b),* $P(o(t), q.R) \leq min(0.1, 0.2) = 0.1$ *in Figure 5.3(c), and* $P(o(t), q.R) \geq 1 - (0.1 + 0.2) = 0.7$ *in Figure 5.3(d).*

In the next subsection, we exploit *Cantelli's inequality* [Mee03] to derive $P^+(A_1)$, $P^+(A_2)$, $P^+(B_1)$, and $P^+(B_2)$ based on the statistics information of $o(t)$.

5.3.2 Statistics Based Filtering Technique

For presentation simplicity, we use a random variable X to denote the location distribution $o(t)$. Following are formal definitions of the *expectation* and the *variance* of X.

Definition 5.4. ***Expectation***, $E(X)$. *We use $E(X)$ to denote the expectation of X, where $E(X).D_i = \sum_{s \in \mathcal{S}} s.D_i \times P(X = s)$.*

Note that $P(X = s)$ denotes the probability that X resides on the state s, and $s.D_i$ is the i-th coordinate value of the state (location) s.

Definition 5.5. ***Variance***, $\sigma^2(X)$. *We use $\sigma_i^2(X)$ to denote the variance of X on each dimensions; that is, $\sigma_i^2(X) = \sum_{s \in \mathcal{S}} (s.D_i - E(X).D_i)^2 \times P(X = s)$.*

Given two values x and y where $x > 0$ and $y > 0$, we use $\delta(x, y)$ to denote a function where $\delta(x, y) = \frac{1}{1+\frac{x^2}{y^2}}$. Following is *Cantelli's inequality* [Mee03], which is demonstrated in Figure 5.4(a).

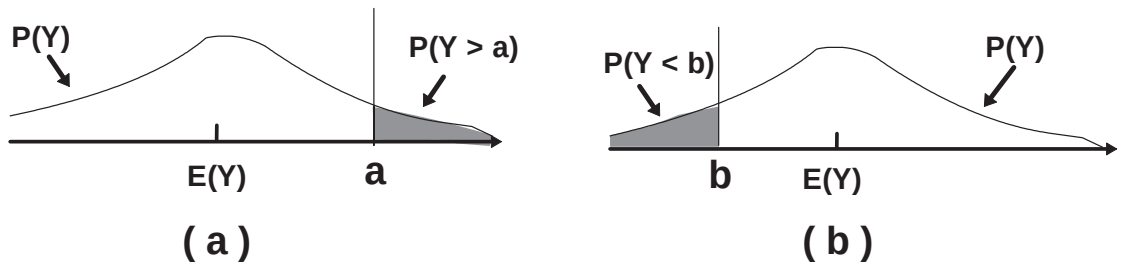


Figure 5.4: Example for Cantelli's Inequality

Lemma 5.1 (Cantelli's Inequality [Mee03])**.** *Suppose that Y is a random variable in one dimensional space with expectation $E(Y)$ and variance $\sigma^2(Y)$, $P(Y \geq a) \leq \delta(a - E(Y), \sigma(Y))$ for any $a > E(Y)$, where $P(Y \geq a)$ denotes the probability of $Y \geq a$.*

By replacing Y with $2E(Y) - Y$, we have a variant of Lemma 5.1 which is illustrated in Figure 5.4(b).

Lemma 5.2. *Suppose that Y is a random variable in one dimensional space with expectation $E(Y)$ and variance $\sigma^2(Y)$, $P(Y \leq b) \leq \delta(E(Y) - b, \sigma(Y))$ for any $b < E(Y)$, where $P(Y \leq b)$ denotes the probability of $Y \leq b$.*

We can come up with $P^+(B_1)$ in Figure 5.3(b) based on Lemma 5.1. And Lemma 5.2 can be used to derive $P^+(A_1)$ in Figure 5.3(a).

Suppose the expectation and the variance of X (i.e., location distribution $o(t)$) are readily available, we can derive the upper bound of $P(X, q.R)$ for *pruning*.

Theorem 5.1. *Suppose that the search region $q.R$ overlaps $X.mbr$ but does not contain $E(X)$. We have $P^+(A_i) = 1$ if $X.mbr$ is contained by $q.R$ on i-th dimension, otherwise $P^+(A_i) = \delta(d_i, \sigma_i(X))$ where d_i denotes the distance between $q.R$ and $E(X)$ on the i-th dimension (e.g., d_1 in Figure 5.3(a)). Then we have $P^+(X, q.R) = min(P^+(A_1), P^+(A_2))$.*

Proof. It is immediate that $P^+(A_i) = P(A_i) = 1$ if $X.mbr$ is contained by $q.R$ on i-th dimension according to the definition of $P(A_i)$. Otherwise, suppose $q.R$ is on the left side or bottom of $E(X)$ on i-th dimension (e.g., A_1 in Figure 5.3(a) and Figure 5.3(c), and A_2 in Figure 5.3(c)), then we have $P^+(A_i) = \delta(d_i, \sigma_i(X))$ according to Lemma 5.2. Similarly, we have $P^+(A_i) = \delta(d_i, \sigma_i(X))$ according to Lemma 5.1 when $q.R$ is on the right side or top of $E(X)$ on i-th dimension. Then we have $P(X, q.R)) \leq min(P(A_1), P(A_2)) \leq min(P^+(A_1), P^+(A_2))$. Thus, the theorem holds. □

With similar rationale, we have the following theorem which can obtain the lower bound of $P(X, q.R)$ for the *validation* of time t.

Theorem 5.2. *Suppose that the search region $q.R$ overlaps $X.mbr$ and contains $E(X)$. We have $P^+(B_i) = 0$ if $X.mbr$ is contained by $q.R$ on i-th dimension, otherwise $P^+(B_i) = \delta(d_i, \sigma_i(X))$ where d_i denote the distance between $E(X)$ and the uncovered area on i-th dimension (e.g., d_1 in Figure 5.3(b)). Then we have $P^-(X, q.R) = 1 - (P^+(B_1) + P^+(B_2))$.*

Statistics Based Filtering. For a given segment $g(o, t_i, t_j)$, the *expectation* and *variance* information are maintained for each time $t \in (t_i, t_j)$. For a given query q, we can obtain $P^-(o(t), q.R)$ and $P^+(o(t), q.R)$ according to Theorem 5.1 and Theorem 5.2. Let $\mathcal{T}(g)$ denote the timestamps within g which satisfy query time constraints, i.e., $\mathcal{T}(g) = [t_i, t_j) \cap [q.s, q.e]$. For each timestamp $t \in \mathcal{T}(g)$, we increase $d^-(g)$ and $d^+(g)$ by one if $P^-(o(t), q.R) \geq \theta$. Otherwise, we increase $d^+(g)$ by one if $P^+(o(t), q.R) \geq \theta$.

Reduce Summary Size. To reduce the space consumption, we may keep the statistics information for a set of consecutive times $\{t_k, t_{k+1}, \ldots, t_l\}$ within a segment. Then instead of keeping the expectation and the variance information for each individual time, we maintain the minimal bounding rectangle of their expectations as well as the maximal variance value on each dimension. Then Theorem 5.1 and Theorem 5.2 can be adopted in a conservative way such that any time $t \in \{t_k, t_{k+1}, \ldots, t_l\}$ can be *pruned* or *validated* at the same time. We omit the details due to the space limitation.

5.3.3 Performance Analysis

Given a segment $g(o, t_i, t_j)$, we assume the location distributions of the objects for time $t \in [t_i, t_j)$ are readily available by applying Markov Chains based techniques in Section 2.4.2. The construction of the statistics based summary can be finished in $\mathcal{O}(n_s \times \Delta_t(g))$ time where n_s is the average number of tuples in $o(t)$. Regarding the

filtering cost, it takes $\mathcal{O}(1)$ time for each time and hence the total cost is $\mathcal{O}(\Delta_t(g))$.

5.4 Partition Based Approach

In this section, we introduce a new filtering approach to build summary for a segment based on both spatial and temporal partitions. Specifically, Section 5.4.1 introduces the motivation of the partition based filtering approach. Details of the technique are presented in Section 5.4.2. We discuss how to effectively construct partition based summary in Section 5.4.3.

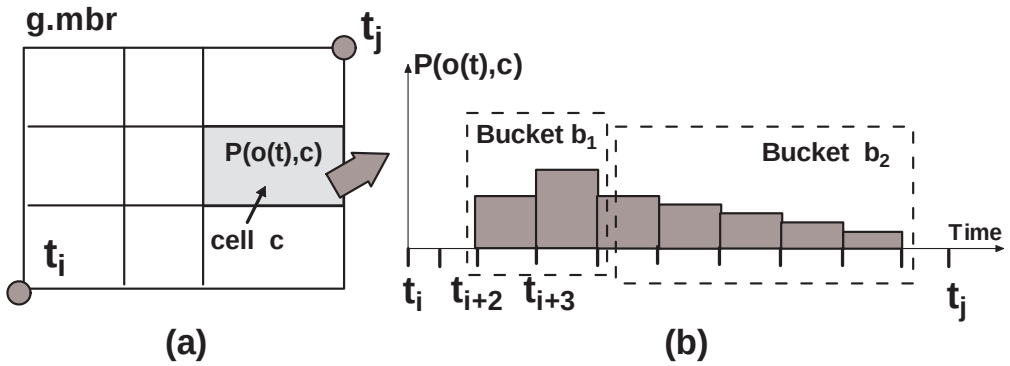


Figure 5.5: Motivation of Partition based Filtering

5.4.1 Motivation

Although statistics based filtering technique is simple and effective, a considerable number of segments will still survive from the filtering phase in our empirical study, because it is difficult to precisely capture a distribution when there are only a few statistics. This motivates us to develop more sophisticated summary technique by partitioning along both spatial and temporal dimensions, so that the filtering performance can be significantly enhanced with a reasonable space overhead.

As shown in Figure 5.5(a), we first partition the minimal bounding rectangle of the segment ($g.mbr$) into a set $\mathcal{S}(g)$ of cells. Then for each cell c, Figure 5.5(b)

shows that there is a function $P(o(t), c)$ which presents the appearance probability of the object o within the cell c for times $t \in (t_i, t_j)$. We can immediately derive $P^+(o(t), q.R)$ and $P^-(o(t), q.R)$ as follows.

$$P^-(o(t), q.R) = \sum_{c \in \mathcal{S}(g) \ \wedge \ q.R \text{ contains } c} P(o(t), c) \tag{5.2}$$

$$P^+(o(t), q.R) = \sum_{c \in \mathcal{S}(g) \ \wedge \ q.R \text{ overlaps } c} P(o(t), c) \tag{5.3}$$

It is infeasible to explicitly keep $P(o(t), c)$ values for all $t \in (t_i, t_j)$. Consequently, we employ a set of buckets to approximate the appearance probability distribution for each cell c. For instance, in Figure 5.5(b) we use two buckets $\{b_1, b_2\}$ to represent $P(o(t), c)$ where the time interval size, maximal and minimal appearance probabilities of a bucket b are denoted by $\Delta_t(b)$, $b.p^+$ and $b.p^-$ respectively. In particular, we have $P^-(o(t), c) = b(t, c).p^-$ and $P^+(o(t), c) = b(t, c).p^+$ where $b(t, c)$ is the corresponding bucket of c at time t. Then, we have

$$P^-(o(t), q) = \sum_{c \in \mathcal{S}(g) \ \wedge \ q.R \text{ contains } c} b(t, c).p^- \tag{5.4}$$

and

$$P^+(o(t), q) = \sum_{c \in \mathcal{S}(g) \ \wedge \ q.R \text{ overlaps } c} b(t, c).p^+ \tag{5.5}$$

5.4.2 Partition Based Filtering

Given a partition based summary of a segment $g(o, t_i, t_j)$, denoted by $\mathcal{S}(g)$, we can come up with an effective computation algorithm to derive lower and upper bounds for $d(g)$ according to Equations 5.4 and 5.5. Algorithm 11 illustrates the details. Specifically, we first retrieve cells in $\mathcal{S}(g)$ which are contained by $q.R$ or overlap $q.R$, denoted by $\mathcal{C}_\cap$ and $\mathcal{C}_\cup$, respectively. Then we attempt to *validate* (Line 7-12) or *prune* (Line 14-18) a time $t \in \mathcal{T}(g)$. Line 8 calculates the lower bound

of $P(o(t), q.R)$ based on Equation 5.4, while Line 15 derives the upper bound according to Equation 5.5. For the given probability threshold θ, we can *validate* a time t if $P^-(o(t), q.R) \geq \theta$ (Line 10). Similarly, t is *pruned* if $P^+(o(t), q.R) < \theta$ (Line 17).

Time Complexity. Let C_r denote the cost to retrieve the cells which are contained by $q.R$ or overlaps $q.R$, and there are n_c cells in $\mathcal{C}_\cup$. Then the total filtering cost is $O(C_r + n_c \times \Delta_t(g))$.

5.4.3 Summary Construction

To effective construct $\mathcal{S}(g)$, we aim to address following three issues in this subsection: *i*) how to generate buckets for a given cell; *ii*) the number of cells assigned for each segment; *iii*) how to generate the cells.

(**i**) **Bucket generation**. As discussed in Section 5.4.1, we cannot afford to keep $P(o(t), c)$ values for all times in $[t_i, t_j)$. Thus, we use B buckets to approximate the probability distribution. Suppose each time t have the same chance to be involved in the filtering phase, the uncertainty introduced by a bucket partition $\mathcal{B}$, denoted by $C(\mathcal{B})$, is as follows.

$$C(\mathcal{B}) = \sum_{t \in (t_i, t_j)} (b(t, c).p^+ - b(t, c).p^-) \tag{5.6}$$

Recall that $b(t, c)$ is the bucket used for the time t. Similar to V-optimal histogram [JKM+98], we come up with the optimal bucket partition $\mathcal{B}^*$ with cost $O(\Delta_t(g)^2 \times B)$ by applying dynamic programming technique.

In our implementation, the number of buckets for each cell is linear to the duration of the segment; that is, $B = \Delta_t(g)/l$ where l is a pre-given constant.

(**ii**) **Number of cells assigned for each segment**. Given two segments g_1 and g_2, we use $Area(g_1)$ ($Area(g_2)$) to denote the area of $g_1.mbr$ ($g_2.mbr$), and n_1 (n_2)

Algorithm 11: ***Partition based Filter***$(\mathcal{S}(g), q, \theta)$

Input : $\mathcal{S}(g)$: partition based summary of g,
q : range search,
θ : probabilistic threshold

Output: $d^-(g)$ and $d^+(g)$

```
1  d⁻(g) := 0; d⁺(g) := 0;
2  C∩ ← cells in S(g) which are contained by q.R;
3  C∪ ← cells in S(g) which overlap q.R;
4  for each time t ∈ T(g) do
5      P⁻(o(t), q.R) := 0;
6      P⁺(o(t), q.R) := 0;
7      for each cell c in C∩ do
8          P⁻(o(t), q.R) := P⁻(o(t), q.R) + b(t, c).p⁻;
9      if P⁻(o(t), q.R) ≥ θ then ;                          // validate
10
11         d⁻(g) := d⁻(g) + 1;
12         d⁺(g) := d⁺(g) + 1;
13     else
14         for each cell c in C∪ do
15             P⁺(o(t), q.R) := P⁺(o(t), q.R) + b(t, c).p⁺;
16         if P⁺(o(t), q.R) ≥ θ then ;                      // prune
17
18             d⁺(g) := d⁺(g) + 1;
19 return d⁻(g) and d⁺(g);
```

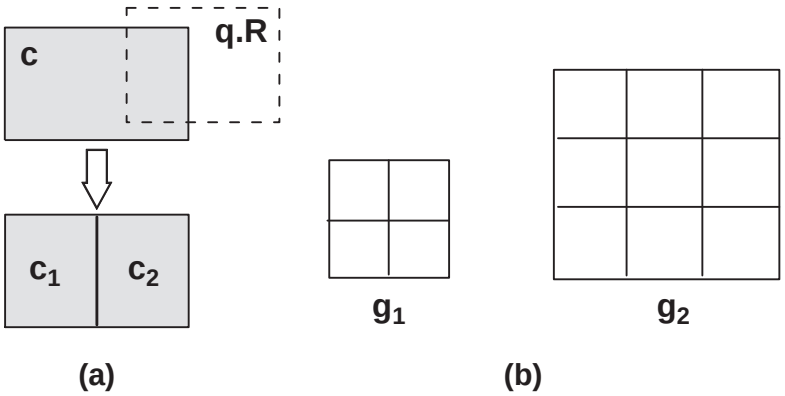


Figure 5.6: Motivation of Resource Allocation

denote the number of cells assigned to g_1 (g_2). At the first glance, we may assign a fixed number of cells to each segment (i.e., $n_1 = n_2$) or fix the area of each cell (i.e., $\frac{n_1}{n_2} = \frac{Area(g_1)}{Area(g_2)}$). However, both strategies are not cost-effective according to our observation below.

With similar spirit to [ZZLL12], we use $P(c) \times W(c)$ to measure the contribution of uncertainty for a cell c, where $P(c)$ is the probability that c overlaps $q.R$ but $q.R$ doesn't contain c, while $W(c)$ denotes the probability mass within the cell (i.e., $W(c) = \sum_{t \in (t_i, t_j)} P(o(t), c)$). As shown in Figure 5.6(a), we can reduce the uncertainty by evenly partitioning a cell c into two parts c_1 and c_2. Assume the probability mass in c is also evenly distributed, now the overall uncertainty cost becomes $2 \times (\frac{P(c)}{2} \times \frac{W(c)}{2}) = \frac{P(c) \times W(c)}{2}$. In Figure 5.6(b), we assume $W(g_1) = W(g_2)$ (i.e., two segments have the same duration length), and two cells from the same segment have the same area and probability mass. Intuitive, for a cost-effective resource allocation strategy, each cell should contribute the same amount of uncertainty; that is, $\frac{P(g_1)}{n_1} \times \frac{W(g_1)}{n_1} = \frac{P(g_2)}{n_2} \times \frac{W(g_2)}{n_2}$. Consequently, we have

$$(\frac{n_1}{n_2})^2 = \frac{Area(g_1)}{Area(g_2)} \tag{5.7}$$

since $\frac{P(g_1)}{P(g_2)} \approx \frac{Area(g_1)}{Area(g_2)}$. For instance, suppose we have $Area(g_1) = 10$ and $Area(g_2) = 40$ in Figure 5.6(b), if 4 cells are assigned to g_1, then 8 cells should goes to g_2 because

$(\frac{4}{8})^2 = \frac{10}{40}$.

(**iii**) **Cell generation**. Now we investigate how to partition the minimal bounding rectangle of a segment g into $m_1 \times m_2$ cells. Following the above argument, the uncertainty cost of a segment g is $\sum_{c \in \mathcal{S}(g)} P(c) \times W(c)$. Because $\sum_{c \in \mathcal{S}(g)} Area(c)$ and $\sum_{c \in \mathcal{S}(g)} W(c)$ are two constants regardless how the cells are generated. This implies that we should have the same $Area(c) \times W(c)$ value for each cell in $\mathcal{S}(g)$ to minimize the uncertainty cost according to Chebyshev Sum Inequality [HLP88]. Nevertheless, it is infeasible to achieve this with a regular grid partition, and hence we resort to a simple heuristic. In particular, for a segment with $m_1 \times m_2$ cells we partition $g.mbr$ into m_i parts with the same probability mass along each dimension i, which can be finished in time $\mathcal{O}(n_s \times \Delta_t(g) + n_s \log(n_s))$ where n_s is the average number of tuples in $o(t)$ for $t \in (t_i, t_j)$.

5.5 Experiment

In this section, we present results of a comprehensive performance study to evaluate the efficiency and scalability of the proposed techniques in the chapter. Following algorithms are evaluated.

- **UST:** The range search techniques proposed in [EKM+12a] where sub-diamonds based filtering technique is employed[1].
- **STA:** Algorithm 10 in Section 5.2 where statistics based filtering technique (Section 5.3) is employed.
- **GRID:** Algorithm 10 in Section 5.2 where partition based filtering technique (Section 5.4) is employed.

[1]The range search with *exists* semantics (i.e., $\eta = 1$) is investigated in [EKM+12a]. Nevertheless, their technique can be easily extended to support range search with $\eta > 1$.

In this work, we use techniques proposed in [EKM+12b] to perform candidate refinement for the above three algorithms.

Datasets. We evaluate our techniques on both synthetic and real datasets using data generator from [EKM+12a, NZE+13] with following steps. We construct a two dimensions state space, consisting of n states, which *uniformly* distributed in the domain $[0, 1]^2$. For each state, we randomly choose several neighbors, and then assign random probability to each connection such that the total probability equals to 1. This builds a directed graph where the vertices represent the states and the edges represent the transition probabilities from one state to another. The graph is stored in a matrix as the transition matrix. To create an uncertain trajectory, we randomly choose one state as the start point, and then apply a directed random walk through the non-zero edges of the graph to get a moving sequence with 100 time steps. The size of time domain is set to $1,000$, and the start time of an uncertain trajectory is randomly chosen between $[1, 900]$. The observations of an object are randomly chosen from the moving sequence. In the experiment, we generate $10,000$ states with a transition matrix. The number of trajectories N varies from $2,500$ to $10,000$ with default value $5,000$. Two subsequent observations' interval size (i.e., **segment duration**) is randomly chosen from 10 to 15 by default. The probabilistic threshold θ varies from 0.1 to 1.0 with default value 0.5, and the duration threshold η varies from 1 to 10 with default value 6. The real datasets are generated from a set of trajectories of taxis in the city of Beijing [YZXS11]. We apply techniques in [NZE+13] to get the state set, transition matrix, and trajectories set, and then randomly choose $10,000$ state with 583 corresponding trajectories to perform the experiment.

Workload. A workload for the range query consists of 1000 queries in our experiments. The center of a query is uniformly chosen from the domain $[0, 1]^2$, its start

time is randomly generated from $[1, 990]$. The query extent, i.e, search region of a query in each dimension varies from 0.05 to 0.25 with default value 0.1, and the duration of a query ($\Delta_t(q)$) varies from 10 to 25 with default value 10.

Same as [EKM+12a], the catalog size of the UST-Tree is set to 10 in the experiments. In STA Algorithm, we compress the statistics information for every 3 consecutive times. Regarding GRID Algorithm, suppose one cell is assigned to an unit area with size 0.03×0.03. Then for each segment g with area $Area(g.mbr)$, c_n cells are assigned where $c_n = \lceil \sqrt{\frac{Area(g.mbr)}{0.03 \times 0.03}} \rceil$ according to Equation 5.7. Moreover, $\lceil \frac{\Delta_t(g)}{5} \rceil$ buckets are generated for each cell where $\Delta_t(g)$ is the duration of the segment (i.e., observation interval size).

All algorithms proposed in this chapter are implemented in standard C++ with STL library and compiled with GNU GCC. Experiments are run on a PC with Intel Xeon 2.40GHz dual CPU and 4G memory running Debian Linux. The disk page size is fixed to $4,096$ bytes. As the refinement phase contributes the dominant cost in three algorithms, we evaluate their performance by measuring the average number of candidate segments refined. The average query response time is also recorded to evaluate the efficiency of the algorithms.

Table 5.2 lists all parameters which may have impacts on our performance study, where default values are in **bold** font. In our experiments, all parameters use default values unless otherwise specified.

Notation	Definition
number of trajectories (N)	2500, **5000**, 7500, 10000
segment duration $\Delta_t(g)$	[**10**, **15**], [15, 20], [20, 25], [25, 30]
probabilistic threshold (θ)	0.1, 0.3, **0.5**, 0.7, 0.9, 1.0
duration threshold (η)	1, 4, **6**, 8, 10
query extent (area of $q.R$)	0.05, **0.1**, 0.15, 0.20, 0.25
query duration $\Delta_t(q)$	**10**, 15, 20, 25

Table 5.2: Parameter Settings

5.5.1 Performance Tuning

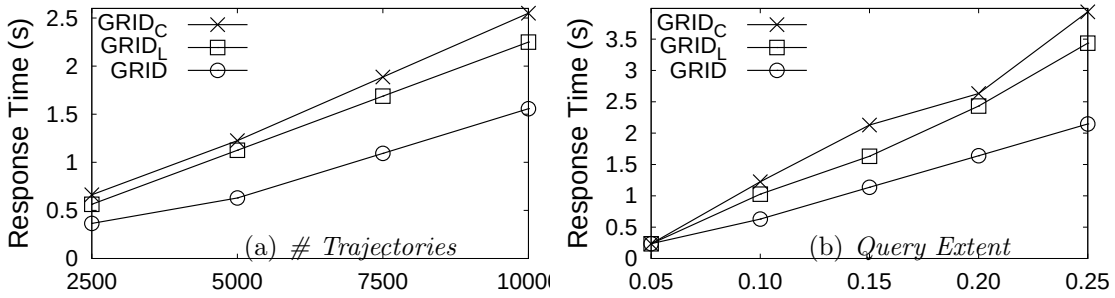


Figure 5.7: Performance Tuning

To evaluate effectiveness of the adaptive resource allocation strategy in Section 5.4.3, we also construct partition based summaries following other two alternative strategies, namely GRID_C and GRID_L respectively. Particularly, GRID_C always allocates 3×3 cells for each segment (i.e., fix the number of cells for each segment), while each cell in GRID_L has the area 0.03×0.03 (i.e., fixed the area of each cell).

Note that summaries constructed in three algorithms have similar summary size under default settings. Nevertheless, Figure 5.7 shows that GRID always outperforms the other two competitors by varying the number of trajectories and query extent. This confirms the effectiveness of our adaptive resource allocation strategy.

5.5.2 Performance Evaluation

Impact of the number of trajectories. Figure 5.8 evaluates the performance of three algorithms on synthetic dataset where the number of trajectories N grows from $2,500$ to $10,000$. With a larger number of trajectories, more trajectories are involved in the computation, thus incurring higher computation cost and more

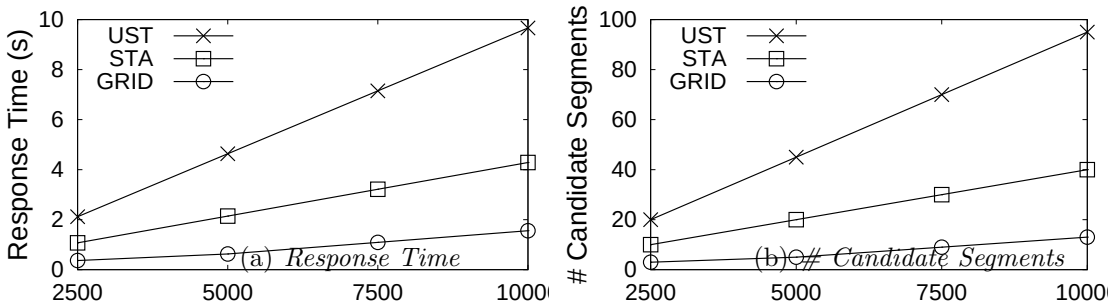


Figure 5.8: Impact of # Trajectories

candidate segments. The response time and the number of candidate segments of STA and GRID grow slowly, yet the performance of UST drops more quickly with the growth of the number of trajectories. It is shown that GRID has the best scalability among three algorithms.

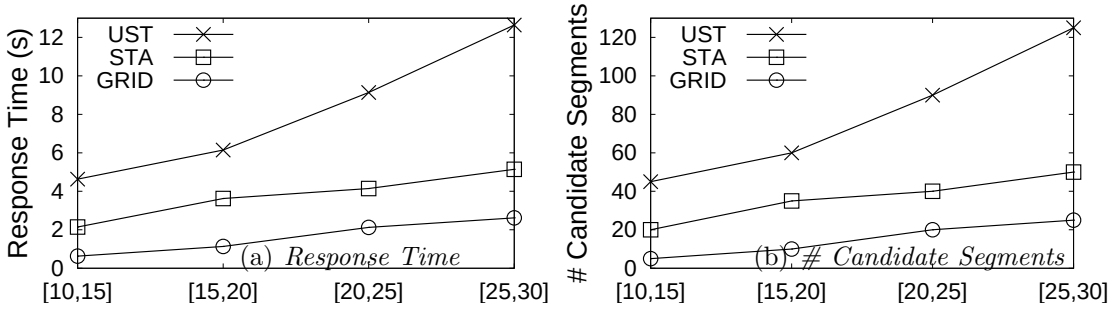


Figure 5.9: Impact of Segment Duration

Impact of segment duration. Figure 5.9 evaluates the impact of the segment duration $\Delta_t(g)$ on three algorithms where $\Delta_t(g)$ is randomly chosen from each bounded interval. The response time and the number of candidate segments are reported for three algorithms. As expected, the performance of UST degrades quickly because it is difficult to find a proper sub-diamond when the segment duration grows. Recall that a sub-diamond $\Diamond$ on a segment $g(o, t_i, t_j)$ need to enforce that the appearance probability of $o(t)$ is bounded by $P(\Diamond)$ regarding *all*

times $t \in [t_i, t_j)$. On the contrary, STA and GRID are much less sensitive to the growth of the segment duration because of the temporal partition; that is, we build summaries for a set of time intervals in g, instead of the whole time interval.

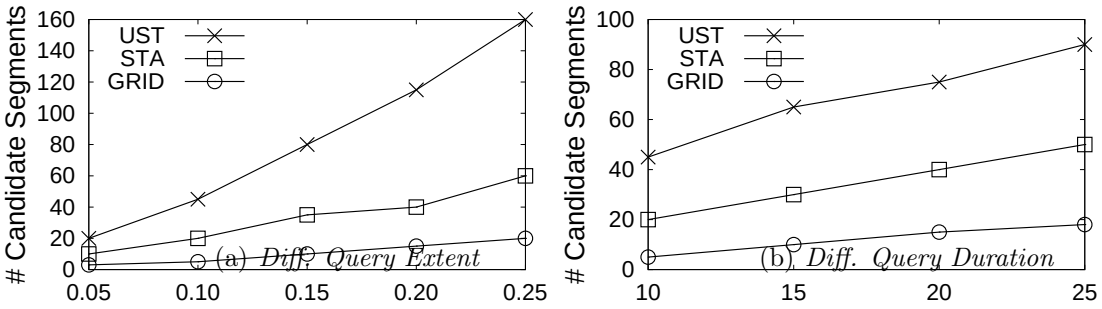


Figure 5.10: Impact of Query Extent and Duration

Impact of query extent and duration. We evaluate the impact of the query extent of $q.R$ as well as the query duration $\Delta_t(q)$ against three algorithms, where the query extent grows from 0.05 to 0.25, and the query duration varies from 10 to 25. With the grow of query extent and query duration, more trajectories are involved in the range search, thus the number of candidate segments increases as expected. Figure 5.10 shows that GRID has the best filtering capability among three algorithms.

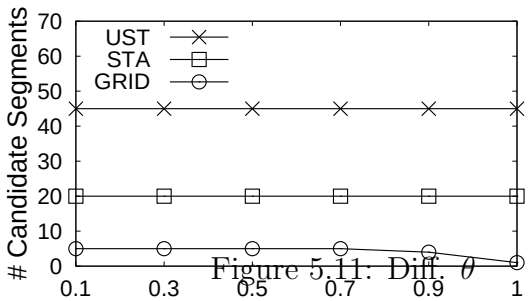


Figure 5.11: Diff. θ

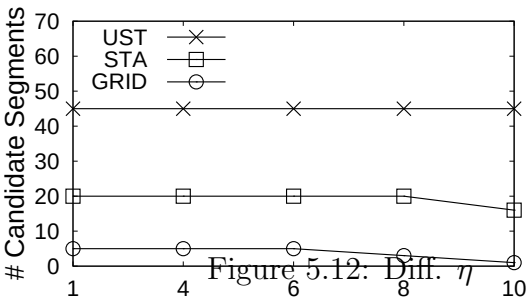


Figure 5.12: Diff. η

Impact of probabilistic threshold. Figure 5.11 investigates the performance of three algorithms as a function of the probabilistic threshold θ which varies from

0.1 to 1. The performance of three algorithms is not sensitive to θ. It is shown that GRID always has the best performance among three algorithms.

Impact of duration threshold. Figure 5.12 reports the number of candidate segments of the algorithms as a function of the duration threshold η which varies from 1 to 10. It is shown that the growth of η does not noticeably affect performance of three algorithms, while GRID always achieves the best performance under all settings. Recall that, when η equal 1, the range search exactly corresponds to the range search with *exists* semantics studied in [EKM+12a].

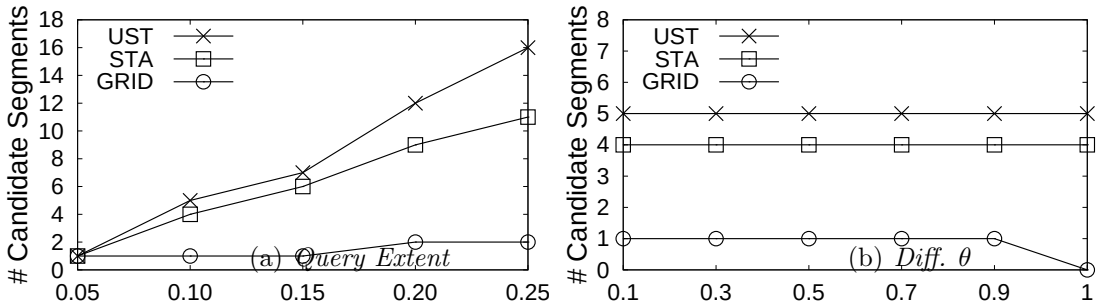


Figure 5.13: Experiments on Real Data

Real data. We also perform experiments on the real-life dataset. Figure 5.13 reports the number of candidate segments against the growth of the query extent and probabilistic threshold θ. Similar trends are observed in Figure 5.13 compared with the experiments on the synthetic data.

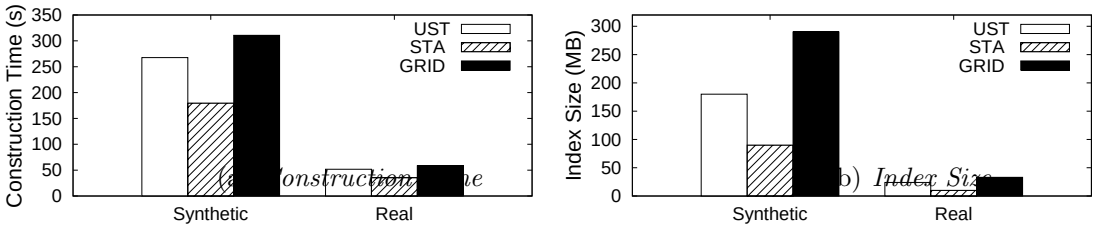


Figure 5.14: Index Construction

Index construction. Figure 5.14 reports the index construction time and the index size of three algorithms on synthetic data and real-life data. It is interesting that STA outperforms UST in term of query response time, while STA also consumes less index size and index construction time. On the other hand, although GRID has the largest index size and construction time, it is cost effective considering of its superior performance compared with other two algorithms.

5.6 Conclusion

To tame the uncertainty of trajectory data collected in a wide spectrum of applications, we have developed effective filtering and query processing techniques to support range search on uncertain trajectories which are modeled by Markov Chains. Particularly, we formally define the problem of range search on uncertain trajectories. Then we introduce an indexing structure where the summaries of the segments are organized by an SS-Tree, as well as a general framework to support range search on uncertain trajectories following the filtering and refinement paradigm. To enhance the filtering capabilities, we develop effective statistics based and partition based filtering techniques. Our comprehensive experiments demonstrate the superior performance of our new techniques compared with existing work.

Chapter 6

Top k Similarity Join

In this chapter, we investigate the top k similarity join problem over multi-valued objects based on a ϕ-quantile distance ($\phi \in (0, 1]$); for example, in sport game, median is the 0.5-quantile, maximum is the 1-quantile, minimum is the smallest quantile (note a quantile ϕ is in $(0, 1]$ and cannot be 0). Regarding the above case, 0.5-quantile is based on players' median performance; 1-quantile is to retrieve the top k similar pairs based on players' worst performance. ϕ-quantile group-base distance, on the other hand, aims at the "best population" (specified by ϕ) regarding the distance of each object pair. Therefore, we study the problem of top k similarity joins over multi-valued objects where the input are two sets of multi-valued objects, and the two types of quantile distances are investigated respectively.

This chapter is organized as follows. Section 6.1 formally defines the problem of top k similarity join over multi-valued objects regarding two types of quantile distance measures, ϕ-quantile distance and ϕ-quantile group-base distance, and provide some necessary background information. In Section 6.2, we introduce the filtering-refinement framework, as well as the data structures utilized in the chapter. Section 6.3 and Section 6.4 present techniques for top k similarity join based on ϕ-

quantile distance and ϕ-quantile group-base distance, respectively. In Section 6.5, we report our experiment results. Some extension of the proposed techniques are discussed in Section 6.6. This is followed by conclusion in Section 6.7.

6.1 Background

We present problem definition and necessary preliminaries in this section. For references, notations frequently used in the chapter are summarized in Table 6.1.

Notation	Definition
$\mathcal{U}$, $\mathcal{V}$	two sets of objects in the join query
U (V)	multi-valued object
E	entry of R-tree
u (v)	instance of U (V) - a point in d-dimensional space
$w(u)$ $(w(S))$	(total) weight of u (the set S)
$d(u, v)$	Euclidean distance between u and v
$d^{lo}(E, E')$	distance lower-bound between E and E'
$d_\phi(U, V)$	ϕ-quantile distance of U and V
$gbd_\phi(U, V)$	ϕ-quantile group-base distance of U and V
$U \times V$	Cartesian product of instances from U to V

Table 6.1: The summary of Notations.

6.1.1 Problem Definition

Multi-valued Object. In our problem definition, an instance of an object U is weighted - weight gives the *representativeness* of an instance in U. For instance, in the sport game example, a game statistic of a player may appear multiple times; consequently a normalized weight (the occurrence of an instance over the total occurrences of all instances) may be used to indicate the representativeness of an instance. Note that the total of such weights in U equals 1.

A multi-valued object U is represented as $\{(u_i, w(u_i))|1 \leq i \leq m\}$ where u_i is a point in a d-dimensional space, $0 < w(u_i) \leq 1$ $(1 \leq i \leq m)$, and $\sum_{i=1}^{m} w(u_i) = 1$.

We use $\mathcal{U}$ and $\mathcal{V}$ to denote two sets of multi-valued objects involved in the join query.

Below we define the ϕ-quantile distance and ϕ-quantile group-base distance between two multi-valued objects.

Quantile. Given a collection S of m elements, each element s_i has a weight $w(s_i)$ where $0 < w(s_i) \leq 1$ and $\sum_{i=1}^{m} w(s_i) = 1$. Let S be sorted increasingly on a search key f - a function; that is, $f(s_i) \leq f(s_j)$ if $i < j$.

Definition 6.1 (ϕ-quantile of S)**.** *Given a ϕ ($0 < \phi \leq 1$), the ϕ-quantile S_ϕ of S is the* **first** *element s_i in the* **sorted** *S on the search key such that $\sum_{j=1}^{i} w(s_j) \geq \phi$.*

ϕ-quantile Distance. For two given objects U and V, there are totally $(|U| \times |V|)$ pairs of instances in $U \times V$ where each pair (u_i, v_j) ($u_i \in U$ and $v_j \in V$) has the weight $w(u_i) \times w(v_j)$, namely $w(u_i, v_j)$. Clearly, $\sum_{u_i \in U, v_j \in V} w(u_i) \times w(v_j) = 1$. The *Euclidean distance* $d(u_i, v_j)$ [1] between u_i and v_j is called the distance of (u_i, v_j). Let $U \times V = \{((u_i, v_j), w(u_i, v_j)) \mid u_i \in U \ \& \ v_j \in V\}$.

Definition 6.2 (ϕ-quantile distance of U and V)**.** *Given a $\phi \in (0, 1]$, let $U \times V$ be sorted increasingly on the search key - the distance $d(u_i, v_j)$ of each element (u_i, v_j). Then, the distance of the ϕ-quantile of $U \times V$ is called the* ϕ-quantile distance *of $U \times V$, denoted by $d_\phi(U, V)$.*

Definition 6.2 states that if (u, v) is the ϕ-quantile of $U \times V$ (i.e., $(U \times V)_\phi = (u, v)$) then $d(u, v)$ is $d_\phi(U, V)$.

Example 6.1. *Regarding the example in Figure 6.1, $|U| = 3$ and $|V| = 2$. Assume that $w(u_1) = \frac{1}{2}$, $w(u_2) = w(u_3) = \frac{1}{4}$; $w(v_1) = w(v_2) = \frac{1}{2}$. Consequently, $U \times V$ consists of the following six pairs sorted on their distances increasingly:*

[1] Note that our techniques developed in this chapter are based on Euclidean distance; nevertheless they can be immediately extended to cover other distance metrics.

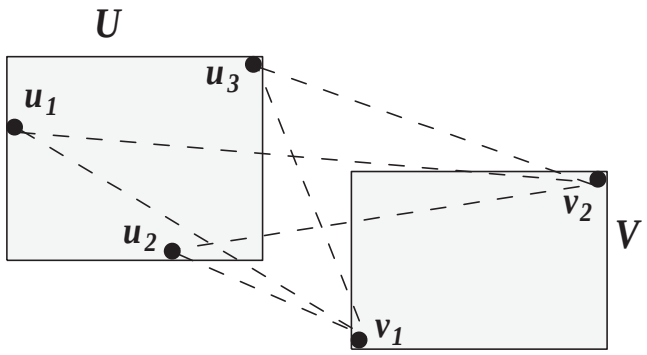


Figure 6.1: Distances between Two Multi-Valued Objects

$U \times V = \{((u_2, v_1), \frac{1}{8}), ((u_3, v_1), \frac{1}{8}), ((u_3, v_2), \frac{1}{8}), ((u_1, v_1), \frac{1}{4}), ((u_2, v_2), \frac{1}{8}), ((u_1, v_2), \frac{1}{4})\}$. *The* 0.2*-quantile distance* $d_{0.2}(U, V)$ *of* U *and* V *is* $d(u_3, v_1)$, $d_{0.5}(U, V)$ *is* $d(u_1, v_1)$, $d_{0.6}(U, V)$ *is also* $d(u_1, v_1)$.

Below we introduce ϕ-quantile group-base distance measure, which is defined based on the *top/best* quantile-population of S.

Definition 6.3 (ϕ-quantile population of S)**.** *Given a* S *and a* $\phi \in (0, 1]$*, a* ϕ*-quantile population* $S_{\phi,P}$ *of* S *is a sub-collection* S' *of* S *such that the total weights of the elements in* S' *is not smaller than* ϕ *and removing any element from* S' *makes the total weights in the remaining sub-collection smaller than* ϕ*.*

Definition 6.4 (ϕ-quantile group-base distance)**.** *Given a* $\phi \in (0, 1]$*, the* ϕ*-quantile group-base distance of* U *and* V *is the minimum total weighted distance among* ϕ*-quantile populations of* $U \times V$*; that is, the minimum value of* $\sum_{(u,v)\in S'} w(u)w(v)d(u, v)$ *with the constraint that* S' *is a* ϕ*-quantile population of* $U \times V$*.*

The ϕ-quantile group-base distance between U and V is denoted as $gbd_\phi(U, V)$.

Example 6.2. *Regarding Example 6.1, let* $\phi = 0.5$*.* $gbd_{0.5}(U, V) = \frac{1}{8}d(u_2, v_1) + \frac{1}{8}d(u_3, v_1) + \frac{1}{8}d(u_3, v_2) + \frac{1}{8}d(u_2, v_2)$ *instead of* $\frac{1}{8}d(u_2, v_1) + \frac{1}{8}d(u_3, v_1) + \frac{1}{8}d(u_3, v_2) + \frac{1}{4}d(u_1, v_1)$.

In fact, there are several 0.5*-quantile populations of* $U \times V$*, including* $\{((u_3, v_1), \frac{1}{8}),\ ((u_2, v_2), \frac{1}{8}),\ ((u_1, v_1), \frac{1}{4})\}$, $\{((u_2, v_1), \frac{1}{8}),\ ((u_2, v_2), \frac{1}{8}),\ ((u_1, v_1), \frac{1}{4})\}$, *etc.*

Note that for a $\phi \in (0, 1]$, Example 6.2 shows that $gbd_\phi(Q, U)$ is not always defined on the set of the "consecutive" smallest distances. In Example 6.2, $\{((u_2, v_1), \frac{1}{8}), ((u_3, v_1), \frac{1}{8}), ((u_3, v_2), \frac{1}{8}), ((u_1, v_1), \frac{1}{4})\}$ is not even a 0.5-quantile population of $U \times V$. In fact, we will show in Section 6.4 that the computation of $gbd_\phi(Q, U)$ is NP-hard.

Problem Statement.

ϕ-quantile Top k Similarity Join. Given a $\phi \in (0, 1]$, two sets of multi-valued objects $\mathcal{U}$ and $\mathcal{V}$ in the d-dimensional space, a ϕ-quantile top k similarity join retrieves k pairs of objects $\mathcal{P}$ from $\mathcal{U} \times \mathcal{V}$ such that for each object pair (U, V) from $\mathcal{P}$, its ϕ-quantile distance $d_\phi(U, V)$ is no greater than the ϕ-quantile distance of object pairs from $\mathcal{U} \times \mathcal{V} - \mathcal{P}$.

ϕ-quantile Group-base Top k Similarity Join. Given a $\phi \in (0, 1]$, two sets of multi-valued objects $\mathcal{U}$ and $\mathcal{V}$ in the d-dimensional space, a ϕ-quantile group-base top k similarity join retrieves k pairs of objects $\mathcal{P}$ from $\mathcal{U} \times \mathcal{V}$ such that for each object pair $(U, V) \in \mathcal{P}$, its ϕ-quantile group-base distance $gbd_\phi(U, V)$ is no greater than the ϕ-quantile group-base distance of object pairs from $\mathcal{U} \times \mathcal{V} - \mathcal{P}$.

6.1.2 Preliminaries

ϕ-quantile Distance Computation. Given a collection S of m elements, each element s_i has a weight $w(s_i)$ where $0 < w(s_i) \leq 1$ and $\sum_{i=1}^{m} w(s_i) \leq 1$. A naive way to compute the ϕ-quantile is to firstly sort S regarding a given search key f, and then scan the sorted list to obtain the ϕ-quantile of S. Clearly, the naive algorithm runs in $O(m \log m)$.

In [CLRS01], an efficient and effective partitioning technique PARTITIONING (S) is proposed to find an element $s \in S$ to divide S into two sub-collections S_1 and S_2 with the following properties:

1. for each $s' \in S_1$, $f(s') \leq f(s)$; and for each $s' \in S_2$, $f(s') \geq f(s)$.

2. $|S_1| \geq \frac{3}{10}m - 6$ and $|S_2| \geq \frac{3}{10}m - 6$.

Using the partitioning technique, in Algorithm 12 we present an iteration-based algorithm to compute a ϕ-quantile when S is not sorted.

Algorithm 12: *QUANTILE (S, ϕ)*

Input : S: a collection of m elements

ϕ: $0 < \phi \leq \sum_{i=1}^{m} w(s_i)$

f: specify a search key

Output: ϕ-quantile of S

```
1 (s, S1, S2) ⟵ PARTITIONING (S);
2 if φ ≤ w(S1) then
3     call QUANTILE (S1, φ);
4 else
5     if φ > w(S1) + w(s) then
6         call QUANTILE (S2, φ − w(S1) − w(s));
7     else
8         return s;
```

In Algorithm 12, $w(S_1)$ denotes the total weights of the elements in S_1. When S has only one element, $S_1 = S_2 = \emptyset$. It is shown in [CLRS01] that the time complexity of PARTITIONING (S) is linear - $O(|S|)$. Consequently, each iteration runs in linear time regarding the current sub-collection size. Recall the property

2 above in PARTITIONING (S). It is immediate that the sizes of sub-collections involved in the iterations in Algorithm 12 are exponentially reduced - at the ith iteration bounded by $((\frac{7}{10})^{i-1}m + c)$ where c is a constant; consequently, the time complexity of Algorithm 12 is **linear** - $O(m)$. The correctness of Algorithm 12 immediately follows from the property 1 of PARTITIONING (S).

Regarding two multi-valued objects U and V, there are totally $|U| \times |V|$ instance pairs. Directly applying the partition based algorithm, computing ϕ-quantile distance between U and V takes $O(|U| \times |V|)$. In [ZLC+10], instances inside one multi-valued object are indexed by an R-tree. Based on the R-tree, pruning techniques are proposed to discard instance pairs which are guaranteed not to be the ϕ-quantile of $U \times V$. In this chapter, we use the pruning techniques enhanced, partition based, linear time complexity algorithm in [ZLC+10] as a black box in computing ϕ-quantile distance between two multi-valued objects.

ϕ-quantile Group-base Distance Computation. It is proved in [ZLC+10] that ϕ-quantile group-base distance computation is an NP-hard problem. We adopt the approximate algorithm in [GJ00] while applying it to computing ϕ-quantile group-base distance. Let $approxgbd_\phi(U, V)$ denote the group distance output by the approximation algorithm, the following theorem is shown in [GJ00].

Theorem 6.1.

$$1 \leq \frac{approxgbd_\phi(U, V)}{gbd_{phi}(U, V)} \leq 2$$

The approximation algorithm runs in $O(mlogm)$ where m is number of instance pairs in $U \times V$. In our experiment, it shows that the approximation algorithm is very accurate in practice.

Conventional Top k Similarity Joins. As the quantile distance computation between two objects is very expensive with the presence of multiple instances, in

this problem, we will apply an R-tree index based top k similarity join algorithm to facilitate the prevention of computing quantile distances between unpromising pairs of multi-valued objects. In [CMTV00], several algorithms are proposed using R-tree based indexes including exhaustive algorithm, recursive algorithm and Heap algorithm. Among all the techniques presented in [CMTV00], Heap algorithm demonstrates a better performance in most experiment settings. The priority query based algorithm in [HS98] is quite similar to Heap algorithm except that Heap algorithm performs a simple pruning before inserting an entry pair into the heap. We adopt the Heap algorithm and develop novel pruning techniques to speed up the computation. Note that our pruning techniques are general enough to be plugged into any R-tree based algorithm for computing conventional top k similarity joins.

6.2 Framework

Our techniques for solving the top k similarity join based on both ϕ-quantile distance and ϕ-quantile group-base distance follow a standard seeding-filtering-refinement framework outlined in Algorithm 13.

Algorithm 13: ***Framework***

- **Phase 1 - Seeding:** Compute the ϕ-quantile distance (ϕ-quantile group-base distance) for each of the k chosen object pairs from $\mathcal{U} \times \mathcal{V}$.
- **Phase 2 - Filtering:** Discard unpromising object pairs from $\mathcal{U} \times \mathcal{V}$.
- **Phase 3 - Refinement:** Determine the final solution for ϕ-quantile top k similarity join (ϕ-quantile group-base top k similarity join).

In the seeding phase, we choose k object pairs and compute their ϕ-quantile distances (ϕ-quantile group-base distances), using the techniques introduced in Section

6.1.2. Let λ_k be the maximal of these k ϕ-quantile distances (ϕ-quantile group-base distances), in the filtering phase, λ_k could be used to prune unpromising object pairs and iteratively updated if necessary. Any k object pairs from $\mathcal{U} \times \mathcal{V}$ could be chosen to compute the ϕ-quantile distance (ϕ-quantile group-base distances) in the seeding phase. Obviously, similar object pairs will lead to smaller λ_k values; and hence better pruning power in the filtering phase. In our framework, to select k object pairs, we first use the mean $\mu(U)$ of the multiple instances for each multi-valued object U from the two given datasets to represent U. $\mu(U) = \sum_{i=1}^{m} w(u_i) \times u_i$ where m is the number of instances in U. Clearly $\mu(U)$ is also in the d-dimensional space. Thus the top k similarity join is converted to join over conventional datasets where each object is a single point in the multi-dimensional space, and we could apply the existing algorithms [CMTV00] to obtain the k most similar pairs from the two (single-valued) datasets. The corresponding k multi-valued object pairs from $\mathcal{U}$ and $\mathcal{V}$ are then chosen to compute the ϕ-quantile distances (ϕ-quantile group-base distances). At this point, we obtain a distance threshold λ_k which will be used in the filtering phase.

Data Structures

In our techniques, we use aggregate R-trees [PKZT01] to index the local instances of each multi-valued object in $\mathcal{U} \cup \mathcal{V}$, and use two statistic information enhanced R-trees (named sR-trees) to globally index the minimum bounding boxes (MBRs) of objects in $\mathcal{U}$ and $\mathcal{V}$, respectively. The local aR-trees and global sR-trees are built to facilitate our filtering techniques.

Local aR-trees. For each multi-valued object $U \in \mathcal{U} \cup \mathcal{V}$, a *local* aR-tree [PKZT01] is built to organize its multiple instances. The aggregate information kept on each intermediate entry is the sum of weights of instances indexed by the entry. Namely, for every intermediate entry E in the local aR-tree, we record the

weight of E as the sum of weights (total weights) of instances having E as an ancestor.

Global sR-trees. We maintain two R-trees on the MBRs of multiple instances of objects in $\mathcal{U}$ and $\mathcal{V}$, respectively. That is, for each object in $\mathcal{U}$, we first obtain the MBR of its multiple instances. Then we build an R-tree on these MBRs. This R-tree is called the *global* R-tree of $\mathcal{U}$. Similarly we build the *global* R-tree for $\mathcal{V}$. Note in a global R-tree, each leaf (data) entry is an MBR of an object.

Suppose an object U has m instances in the d-dimensional space, $u_1, u_2, ..., u_m$ with the weights $w(u_1), w(u_2), ..., w(u_m)$, respectively.

Definition 6.5 (Mean μ)**.** *The mean of U, denoted by $\mu(U)$, is $\sum_{i=1}^{m} w(u_i) \times u_i$.*

Note that $\mu(U)$ is in the d-dimensional space. For $1 \leq i \leq d$, $\mu_i(U)$ denotes the i-th coordinate of $\mu(U)$.

Definition 6.6 (Variance σ^2)**.** *For $1 \leq i \leq d$, $\sigma^2(U) = \sum_{j=1}^{m} w(u_j)(u_{j,i} - \mu_i(U))^2$ where each $u_{j,i}$ denotes the i-th coordinate value of u_j.*

In each of the leaf (data) entry of the global R-tree, besides the MBR information of each object, we also keep the above statistic information. And the global R-tree is called a statistic R-tree, denoted by sR-tree. Remind that two sR-tree are built for the multi-valued object sets $\mathcal{U}$ and $\mathcal{V}$, respectively.

6.3 ϕ-Quantile Top k Similarity Join

We present our techniques for ϕ-quantile top k similarity join for a given $\phi \in (0, 1]$ in this section. We first present novel distance, statistic and weight based pruning techniques. Then, we integrate the proposed pruning techniques into the overall join algorithm based on the Heap Algorithm in [CMTV00].

6.3.1 Pruning Techniques

When introducing the pruning techniques, we assume that we have an entry pair (E_U, E_V) from the join processing where E_U (E_V) is an entry from the global sR-tree of $\mathcal{U}$ ($\mathcal{V}$). E_U (E_V) could be either intermediate or leaf (data) entry. The way to access entries from the two global sR-trees will be introduced in Section 6.3.2.

Distance based Pruning. The first pruning rule is based on the distance between two entries in the join processing obtained from intermediate or leaf entries of two global sR-trees.

Pruning Rule 1. Let $d^{lo}(E_U, E_V)$ denote the minimum distance between the MBRs of two entries E_U and E_V. If $d^{lo}(E_U, E_V) \geq \lambda_k$, then (E_U, E_V) can be pruned, namely, all entry pairs in $E_U \times E_V$ can be pruned.

Complexity. Computing the minimum distance between two MBRs takes $O(d)$ time. The complexity of Pruning Rule 1 is constant once d is fixed.

Statistic based Pruning. The second pruning technique utilizes the statistic information kept in the global sR-tree, as introduced in Section 6.2. The main idea is based on the current distance threshold λ_k, to derive a value α such that the α-quantile distance between an object pair (U, V) is not smaller than λ_k. If $\alpha < \phi$, we can safely prune (U, V). We first introduce the *Cantelli's inequality* [Mee03] which is employed in Pruning Rule 2.

$$\delta(x, y) = \begin{cases} \dfrac{1}{1 + \frac{x^2}{y^2}}, y \neq 0 \\ 1, x = 0 \text{ and } y = 0 \\ 0, x \neq 0 \text{ and } y = 0 \end{cases}$$

Theorem 6.2 (Cantelli's Inequality [Mee03])**.** *Suppose that t is a random variable in 1-dimensional space with mean $\mu(t)$ and variance $\sigma^2(t)$, $Prob(t - \mu(t) \geq a) \leq \delta(a, \sigma(t))$ for any $a \geq 0$, where $Prob(t - \mu(t) \geq a)$ denotes the probability of*

$t - \mu(t) \geq a$.

Note that Theorem 6.2 extends the original Cantelli's Inequality [Mee03] to cover the case when $\sigma = 0$ and/or $a = 0$. The following theorem is proved in [LZZC11] and provides an upper-bound for $Prob(t \leq b)$ when $b \leq \mu$.

Theorem 6.3. *Assume that* $0 \leq b \leq \mu(t)$. *Then,* $Prob(t \leq b) \leq \delta(\mu(t) - b, \sigma(t))$.

Proof. Let $t' = 2\mu(t) - t$. It can be immediately verified that $\sigma^2(t') = \sigma^2(t)$ and $\mu(t) = \mu(t')$. Applying Cantelli's Inequality on t', the theorem holds. □

Now we generalize the above observations into our statistic based pruning rule. As shown in Figure 6.2, for two object entries (U, V) stored in the leaf/data entries of global sR-tree of $\mathcal{U}$ and $\mathcal{V}$, along the i-th dimension $(1 \leq i \leq d)$, e.g., the horizontal dimension in Figure 6.2, we locate two lines m and n vertical to the i-th dimension and with distance λ_k between m and n. Denote U_i (V_i) as the coordinate value of U (V) along the i-th dimension. The line $U_i = m$ $(V_i = n)$ divides the MBR of U (V) into two parts, denoted as U_1 and U_2 (V_1 and V_2), as shown in Figure 6.2. Assume $\mu_i(U) < \mu_i(V)$. Remind that λ_k is the current distance threshold.

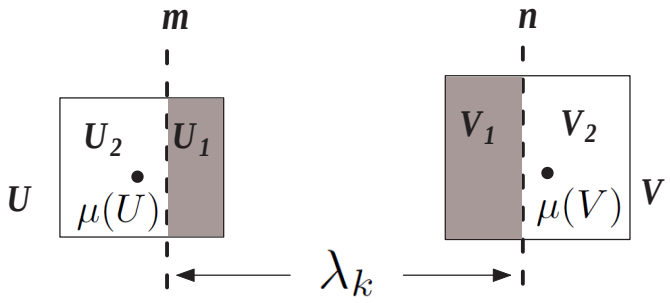


Figure 6.2: Statistic based Pruning

The intuition of the statistic based pruning technique is as follows: along each dimension i, based on Theorem 6.3, we derive an upper bound of the sum of weights in the shaded areas of the MBRs of U_1 and V_1, respectively, denoted as

$W_i^{up}(U_1)$ and $W_i^{up}(V_1)$. Clearly, we can claim that instance pairs from $U_2 \times V_2$ can not have distance smaller than λ_k. Denote the sum of weights in U_2 and V_2 as $W_i(U_2)$ and $W_i(V_2)$, respectively. Obviously, $W_i(U_2) \geq 1 - W_i^{up}(U_1)$, and $W_i(V_2) \geq 1 - W_i^{up}(V_1)$. Thus, using $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$, we can identify a value α such that the α-quantile distance between U and V is not smaller than λ_k. Next we present the monotonic property of quantile distance.

Theorem 6.4 (Monotonicity of Quantile Distance)**.** *Given two multi-valued objects U and V, $\alpha, \phi \in (0, 1]$, if $\alpha < \phi$, then $d_\alpha(U, V) \leq d_\phi(U, V)$.*

Proof. The theorem immediately holds based on the definition of quantile distance in Definition 6.2. □

Based on Theorem 6.4, once we identify the value α such that the α-quantile distance between U and V is larger than λ_k, if $\alpha < \phi$, then we can claim the ϕ-quantile distance between U and V cannot be smaller than λ_k. In this way (U, V) can be pruned based on the statistic information kept in the global sR-tree only without accessing the local aR-trees of U and V.

Pruning Rule 2. Given an object pair (U, V) $(U \in \mathcal{U}, V \in \mathcal{V})$. For a dimension i $(1 \leq i \leq d)$, without lose of generality, assume $\mu_i(U) < \mu_i(V)$. If 1 - (1 - $\delta(m - \mu_i(U), \sigma_i(U))) \times (1 - \delta(\mu_i(V) - n, \sigma_i(U))) < \phi$, (U, V) can be pruned.

Proof. For the i-th $(1 \leq i \leq d)$ dimension, based on Theorem 6.3, we obtain the upper bound of the sum of weight of instances in the shaded area U_1 of the MBR of U as $W_i^{up}(U_1) = Prob(U_i \geq m) \leq \delta(m - \mu_i(U), \sigma_i(U))$. Similarly we get $W_i^{up}(V_1) = Prob(V_i \leq n) \leq \delta(\mu_i(V) - n, \sigma_i(V))$. Since the instance pairs from $U_2 \times V_2$ cannot have distance smaller than λ_k, we have $\alpha \leq 1 - (1 - W_i^{up}(U_1)) \times (1 - W_i^{up}(V_1)) \leq 1 - (1 - \delta(m - \mu_i(U), \sigma_i(U))) \times (1 - \delta(\mu_i(V) - n, \sigma_i(U)))$. Together with Theorem 6.4, the pruning rule is correct. □

Once we obtain an object pair (U, V) from the join processing, we apply Pruning Rule 2 based on the statistic information kept in the global sR-trees before accessing the local aR-trees of U and V. If we encounter a dimension i such that 1 - (1 - $W_i^{up}(U_1)) \times (1 - W_i^{up}(V_1)) < \phi$, the pruning stops and the object pair (U, V) is discarded. As shown in Figure 6.2, after selecting line m along the i-th dimension of U, line n for V is also fixed regarding the current λ_k. We apply the equality principle in determining the position of m and n; namely, the center of m and n is the same as the center of $\mu_i(U)$ and $\mu_i(V)$. Based on Theorem 6.3, we obtain $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$ in constant time.

Complexity. If $W_i^{up}(U_1)$ and $W_i^{up}(V_1)$ are derived based on Theorem 6.3, the time complexity of Pruning Rule 2 is $O(d)$.

Weight based Pruning. The following pruning rule incorporates both weight and distance information. The instances of a multi-valued object are investigated by accessing the local aR-trees. Consider an object entry pair (U, V). If (U, V) is not pruned by Pruning Rule 1 and 2, we explore the instances information of the objects by accessing their local aR-trees. We traverse the local aR-trees of two objects U and V synchronously. At level i, we *trim* object V using the current distance threshold λ_k, and retain only the entries in V with minimum distance to U not larger than λ_k. We record the entries as $\gamma_{V,i}$. Formally, $\gamma_{V,i} = \{E \in L_i(V), d^{lo}(U, E) \leq \lambda_k\}$, where $L_i(V)$ denotes all remaining entries (i.e., not trimed in higher levels) in the local aR-tree of V at the i-th level. Similarly, we obtain $\gamma_{U,i}$. If the multiplication of the weights of $\gamma_{V,i}$ and $\gamma_{U,i}$ is smaller than ϕ, the object pair (U, V) can be pruned as the ϕ-quantile distance between U and V must be larger than λ_k.

Pruning Rule 3. If $\sum_{e \in \gamma_{U,i}} W(e) \times \sum_{e \in \gamma_{V,i}} W(e) < \phi$, the object pair (U, V) can be discarded.

Proof. From the definition of ϕ-quantile distance, it is immediate that if $\sum_{e \in \gamma_{U,i}} W(e) \times \sum_{e \in \gamma_{V,i}} W(e) < \phi$, then $d_\phi(U, V) > \lambda_k$. □

Example 6.3. *As shown in Figure 6.3, at the i-th level, the local aR-tree of object U has two entries U_1 and U_2, local aR-tree of V also has two entries V_1 and V_2. The current threshold λ_k is as illustrated. Using λ_k, we* trim *the MBR of V and only entry V_1 has minimum distance to U smaller than λ_k; thus, $\gamma_{V,i} = \{V_1\}$. Similarly, $\gamma_{U,i} = \{U_2\}$. If $W(U_2) \times W(V_1) < \phi$, the object pair (U, V) could be pruned.*

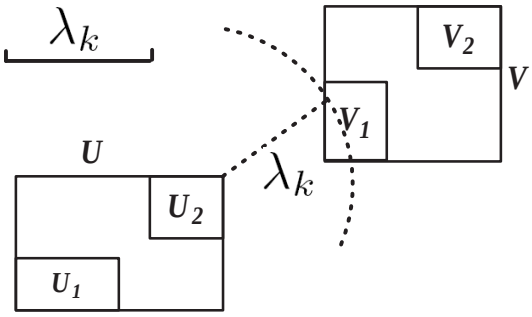


Figure 6.3: Weight based Pruning

By applying Pruning Rule 3, we can avoid accessing all instance pairs of $U \times V$, and seek to stop on intermediate levels of the local aR-trees of U and V. Note the traversal of two aR-trees is in a synchronous fashion and level-by-level from the root node. If one aR-tree reaches leaf nodes first, it stays in leaf level while the other one keeps traversing till its leaf level. As a by-product, if (U, V) cannot be pruned using Pruning Rule 3, we call the ϕ-quantile distance computation algorithm in [ZLC+10] with the instance pairs from $\gamma_{U,i} \times \gamma_{V,i}$ only where i is the leaf (instance) level. Clearly, the algorithm still outputs correct ϕ-quantile distance as the distance of the pruned instance pairs are larger than λ_k based on the definition of $\gamma_{U,i}$ and $\gamma_{V,i}$ for level i.

An exceptional case of Pruning Rule 3 is that we obtain an entry pair (E_U, E_V) from the join processing, one is an object entry while the other is an intermediate

entry. Assume E_U is the object entry of U and E_V is the intermediate entry. Pruning Rule 3 could still be applied to (U, E_V) with the following modifications:

1) We access the local aR-tree of U only and at each level i, record $\gamma_{U,i}$ as the entries in U with minimum distance to E_V not larger than λ_k.

2) if $\sum_{e \in \gamma_{U,i}} W(e) < \phi$, the entry pair (U, E_V) could be pruned. Namely, the object pair of U and any object indexed in E_V must have a ϕ-quantile distance greater than λ_k.

Complexity. Assume the average number of entries at level i of the local aR-trees of multi-valued objects is N_i, then clearly the complexity of Pruning Rule 3 is $O(N_i)$ at each level. The worst case complexity of using Pruning Rule 3 is $O(|U| \times |V|)$, namely no entries are pruned at intermediate entries and we need to access all instance pairs. However, in practice, as shown in Section 6.5, Pruning Rule 3 is very effective and saves CPU costs significantly. Note that in Pruning Rule 3 we trim the entries at each level of local aR-trees of U and V using λ_k instead of considering the combination of all pairs of entries at each level. This is because trim based pruning is more efficient compared with combining all pairs (time complexity $O(N_i^2)$) and also trim based pruning is very effective in practice.

6.3.2 Overall Join Algorithm

The overall join algorithm is adopted from the Heap Algorithm in [CMTV00] as it is both efficient and easy to implement in real applications. We adjust the algorithm to deal with multi-valued objects. Given $\phi \in (0, 1]$, two multi-valued objects sets $\mathcal{U}$ and $\mathcal{V}$, Algorithm 14 illustrates the top k similarity join processing. A minheap H is maintained according to the minimum distance between two entry pairs of the two global sR-trees $R_{\mathcal{U}}$ and $R_{\mathcal{V}}$ indexing $\mathcal{U}$ and $\mathcal{V}$, respectively. H is initialized

with the pair of root nodes of $R_{\mathcal{U}}$ and $R_{\mathcal{V}}$.

The algorithm differentiates three cases based on whether the entries are object entries or not. If both are intermediate entries (Line 5), we expand all the children pairs and insert into heap H the pairs which survive from Pruning Rule 1 (Line 7). If one of the entries is an intermediate entry and the other is an object entry (Line 9), Pruning Rule 1 and 3 will be applied first (Line 10) before expanding the children pairs. We apply all three Pruning Rules on object pairs (Line 13), and if an object pair is survived from pruning, the ϕ-quantile distance is computed; the top k results and λ_k are updated if necessary. Note that even from the root node pair we only insert entry pairs into H if they are not pruned by Pruning Rule 1, it is still necessary to check Pruning Rule 1 (Line 10 and Line 13) since the distance threshold λ_k dynamically changes.

Correctness. Based on the correctness of the three pruning rules, it can be immediately shown that Algorithm 14 is correct.

Discussions. The techniques proposed in this section could be immediately extended to support self-join (i.e., we compute top k similar pairs from one data set $\mathcal{U}$) and threshold base similarity join over multi-valued objects. We omit the details due to space limits.

6.4 ϕ-Quantile Group-base Top k Similarity Join

Our techniques for join processing based on *ϕ-quantile group-base distance* also follow the seeding-filtering-refinement framework in Algorithm 13. In the seeding phase, k object pairs are selected based on the distance between weighted centroid, then corresponding ϕ-quantile group-base distances are computed using the approximation algorithm in [GJ00]. The largest among these k distance values is

Algorithm 14: ***Top k Similarity Join Processing***

Input : $R_{\mathcal{U}}$, $R_{\mathcal{V}}$, k, ϕ

Output: k object pairs from $\mathcal{U} \times \mathcal{V}$ with smallest ϕ-quantile distances

```
1  H = (root(R_U), root(R_U)) if not PRUNED1(root(R_U), root(R_U));
2  while H is not empty do
3      (E_U, E_V) = H.top();
4      H. pop();
5      if E_U and E_V are both intermediate entries then
6          for each children pair (C_{E_U}, C_{E_V}) from E_U × E_V do
7              if not PRUNED1(C_{E_U}, C_{E_V}) then
8                  insert (C_{E_U}, C_{E_V}) into H;
9      else if one of E_U and E_V is an object entry then
10         if not PRUNED1(E_U, E_V) and not PRUNED3(E_U, E_V) then
11             Lines 5 - 8;
12     else                                  /* both E_U and E_V are object entries */
13         if not PRUNED1(E_U, E_V) and not PRUNED2(E_U, E_V) AND not
           PRUNED3(E_U, E_V) then
14             Compute φ-quantile distance between E_U and E_V;
15             if d_φ(E_U, E_V) < λ_k then
16                 Update λ_k and current k most similar pairs;
```

thus utilized as the distance threshold γ_k.

Below we first present the novel and efficient pruning techniques, followed by the overall join processing algorithm based on ϕ-quantile group-base distance metrics.

6.4.1 Pruning Techniques

We assume that we have an entry pair (E_U, E_V) from the join processing where E_U (E_V) could be either an intermediate entry from the global sR-tree of U (V), or a data entry from the local aR-tree of U (V). The following pruning rule is immediate based on the definition of ϕ-quantile group-base distance between two multi-valued objects.

Pruning Rule 4. If $\phi \times d^{lo}(E_U, E_V) \geq \gamma_k$, then (E_U, E_V) can be pruned, namely, all entry pairs in $E_U \times E_V$ can be pruned.

Proof. For each object pair (U, V) where E_U is an ancestor of U and E_V is an ancestor of V, the total weight for any ϕ-quantile population S of $U \times V$ is not smaller than ϕ and the distances of all instances pairs in S are not smaller than γ_k. Thus the pruning condition holds. □

Complexity. Computing the minimum distances between E_U and E_V takes $O(d)$ time, thus the complexity of Pruning Rule 4 is constant once d is fixed.

Note that given U and V, the instance group with the total weighted distance $gbd_\phi(U, V)$ may spread in many different entries of U and V, it is not always possible to trim many entries from the local aR-trees as we do for ϕ-quantile similarity join processing in Pruning Rule 2 and Pruning Rule 3 . The next pruning rule further explores the instances distribution information inside multi-valued objects using the local aR-trees. Before presenting the next pruning technique, we differentiate two cases, 1) both E_U and E_V are data entries, i.e., pointing to a multi-valued

object; and 2) one of them is an intermediate entry in the global sR-tree and the other is a data entry. Note that Case 2) occurs when the two global sR-trees are of different heights.

Considering Case 1) first. Since both E_U and E_V are data entries, we load the corresponding local aR-trees, respectively. The pruning rule is conducted in a level-by-level fashion between the two local aR-trees starting from the root node pairs. Let $L_{U,k}$ and $L_{V,k}$ denote all entries at level k of the local aR-tree of U and V, respectively. Clearly there are overall $|L_{U,k}| \times |L_{V,k}|$ entry pairs. We denote these entry pairs as $L_k = \{(E_U, E_V)_1, \ ... \ (E_U, E_V)_{|L_{U,k}|\times|L_{V,k}|}\}$. Without lose of generality, we assume these entry pairs are sorted in decreasing order based on the minimal distance between the corresponding two MBRs, namely, $d^L(E_U, E_V)_{i_1} \leq d^L(E_U, E_V)_{i_2}$ if $i_1 < i_2$. Let $(E_U, E_V)_j$ denote the ϕ-quantile of L_k where the search key is the minimum distance of two MBRs in each entry pair and the weight of each pair is the multiplication of the weights of the two entries involved in the pair. The intuition of the following pruning rule is to relax the ϕ-quantile distance between U and V to obtain a lower bound of $gbd_\phi(U, V)$.

Pruning Rule 5 (Case 1). Two Multi-valued objects U and V can be pruned if at a level k of the local aR-trees of U and V,

$$(\phi - \sum_{i=1}^{j-1} w((E_U, E_V)_i))d^L(E_U, E_V)_j + \sum_{i=1}^{j-1}(w((E_U, E_V))_i \times d^L(E_U, E_V)_i) \leq \gamma_K$$

Notice that in Pruning Rule 5 (Case 1), if the numbers of instances of U and V are different, it is possible that the two local aR-trees are of different hight. In such scenarios, if one aR-tree reaches the leaf (instance) level first, then the other tree keeps traversing until it also reaches the leaf level.

In Case 2), one of the entry is an intermediate entry from the global sR-tree while the other is a data entry. Without loss of generality, we assume E is an intermediate entry from the sR-tree of object U, the pruning rule is conducted by

accessing the local aR-tree of object V. Suppose that $L_k = \{E_i | 1 \leq i \leq l\}$ consists of all entries at the level k of the local aR-tree of object V and assume that L_k is sorted in the increasing order based on $d^L(E, E_i)$; namely, $d^L(E, E_{i_1}) \leq d^L(E, E_{i_2})$ if $i_1 < i_2$. Let E_j denote the ϕ-quantile of L_k according to the search key $d^L(E, E_i)$ and the weight $w(E_i)$ of each entry $E_i \in L_k$.

Pruning Rule 5 (Case 2). The object pairs $E \times V = \{(U, V) | E \textit{ is the ancestor of } U\}$ could be pruned if at a level k of the local aR-tree of V,

$$(\phi - \sum_{i=1}^{j-1} w(E_i)) d^L(E, E_j) + \sum_{i=1}^{j-1} (w(E_i) \times d^L(E, E_i)) \leq \gamma_K$$

Proof. We prove Case 1) first and Case 2) could be proved in a similar way. Remind that L_k denotes the sorted entry pairs from U and V at k-th level of the corresponding aR-trees where the sorting is conducted according to the minimal distance of MBRs of entry pairs, and $(E_U, E_V)_j$ denotes the ϕ-quantile of L_k. For entry pairs before $(E_U, E_V)_j$ in L_k, we obtain the weighted distance of minimal MBR distances, $\sum_{i=1}^{j-1} (w((E_U, E_V))_i \times d^L(E_U, E_V)_i)$. Deducting this part of weights in ϕ (i.e., $\phi - \sum_{i=1}^{j-1} w((E_U, E_V)_i)$), the remaining part is weighted by the minimal distance of the ϕ-quantile entry pair $(E_U, E_V)_j$. The sum of these two parts is not smaller than the weighted distance in any ϕ-quantile population of $U \times V$. □

Complexity. Remind that $|L_{U,k}|$ and $|L_{V,k}|$ denote the number of entries in the k-th level of the local aR-tree of U and V, respectively. The time complexity for Pruning Rule 5 Case 1) is $O(|L_{U,k}| \times |L_{V,k}| \times log(|L_{U,k}| \times |L_{V,k}|))$ since we need to sort the entry pairs first. Similarly, the time complexity for Pruning Rule 5 Case 2) is $O(|L_k| log |L_k|)$ where $|L_k|$ refers the number of entries at the k-th level of local aR-tree of V.

6.4.2 Overall Join Algorithm

Join processing based on group-base ϕ-quantile distance metrics also follows the Heap Algorithm in [CMTV00]. Based on Algorithm 14, the following changes are made.

- Output object pairs with smallest approximate ϕ-quantile group-base distance.
- In Line 7, PRUNED 1 is changed to PRUNED4.
- In Line 10, PRUNED 1 and PRUNED 3 are changed to PRUNED 4 and PRUNED 5.
- In Line 13, PRUNED 1, PRUNED 2 and PRUNED 3 are changed to PRUNED 4 and PRUNED 5.
- In Line 14, compute $approxgbd_\phi(E_U, E_V)$.
- In Line 15, $d_\phi(E_U, E_V)$ is changed to $approxgbd_\phi(E_U, E_V)$.

Accuracy Guarantee. Due to the hardness of computing ϕ-quantile group based distance, our techniques for ϕ-quantile group-base similarity join yield approximate results with the following accuracy guarantee.

Theorem 6.5. *For $1 \leq i \leq k$, assume that (U_i, V_i) denotes the ith most similar object pair in the exact ϕ-quantile group-base similarity join, (U_i', V_i') denotes the top-ith most similar object pair returned by our algorithms. Then, $gbd_\phi(U_i, V_i) \leq approxgbd_\phi(U_i', V_i') \leq 2gbd_\phi(U_i, V_i)$. Namely, our algorithm has an approximation ratio of 2.*

Proof. Following the proof of Pruning Rule 4 and 5, it is immediate that if an object pair (U, V) is pruned by these two pruning rules, then $gbd_\phi(U, V) \geq \gamma_k$. From Theorem 6.1, it follows that $gbd_\phi(U_i, V_i) \leq approxgbd_\phi(U_i', V_i') \leq 2gbd_\phi(U_i, V_i)$. □

Theorem 6.5 states that the i-th ϕ-quantile group-base distance output by our algorithm is bounded by $gbd_\phi(U_i', V_i')$ and $2gbd_\phi(U_i', V_i')$. Our experimental results show that the algorithm is quite accurate in practice and the error is much smaller.

6.5 Experiment

We report a thorough performance evaluation on the efficiency of proposed techniques and effectiveness of pruning rules. In particular, we implement and evaluate the following techniques.

Join : Techniques presented in Section 6.3 to compute the top k similarity join based on ϕ-quantile distance ($\phi \in (0, 1]$), with Pruning Rule 1, 2 and 3 applied.

P12 : **Join** algorithm but with Pruning Rule 1 and 2 only.

P1 : **Join** algorithm but with Pruning Rule 1 only.

KNN : Baseline algorithm for **Join** by using KNN processing over multi-valued objects in [ZLC+10]. For each object $U \in \mathcal{U}$, we compute its KNN in $\mathcal{V}$ based on ϕ-quantile distance, and then select k most similar pairs based on the union of KNN results.

G-Join : Techniques presented in Section 6.4 to compute the top k similarity join based on group-base ϕ-quantile distance ($\phi \in (0, 1]$), with Pruning Rule 4 and 5 applied.

P4 : **G-Join** algorithm but with Pruning Rule 4 only.

G-KNN : Baseline algorithm for **G-Join** by using group-base quantile KNN processing over multi-valued objects in [ZLC+10]. For each object $U \in \mathcal{U}$, we compute its KNN in $\mathcal{V}$ based on ϕ-quantile group-base distance, and then select k most similar pairs based on the union of KNN results.

All algorithms are implemented in C++ and compiled by GNU GCC. Experiments are conducted on PCs with Intel Xeon 2.4GHz dual CPU and 4G memory under Debian Linux. Our experiments are conducted on both real and synthetic datasets.

Real dataset is extracted from NBA players' game-by-game statistics (http://www.nba.com), containing 339,721 records of 1,313 players. Each player is treated as a multi-valued object where the statistics (score, assistance, rebound) of a player per game is treated as an instance with the equal weight (normalized).

Synthetic datasets are generated using the methodologies in [BKS01] regarding the following parameters. Dimensionality d varies from 2 to 5 with default value 3. Data domain in each dimension is $[0, 1]$. Number n of objects varies from $5,000$ to $15,000$ with default value $5,000$. Number m of instances per object follows a uniform distribution in [1, $\mathcal{M}$] where $\mathcal{M}$ varies from 100 to 800 with the default value 200. The value K varies among 5, 10, 15, 20 and 25 with default value 10. The average length of object MBRs follows a *uniform* distribution spreading over [0, h] where h varies from 0.02 to 0.10 with default value 0.02 (i.e., 2% of the edge length of the whole data space).

Centers of objects (objects' MBRs) follow either *uniform*, *normal* or *anti-correlated* distribution. Locations of instances in an object follow *uniform* or *normal* distribution. Weights assigned to each instance follow *uniform* or *normal* distribution. Table 6.2 summarizes the parameters used in our experiment

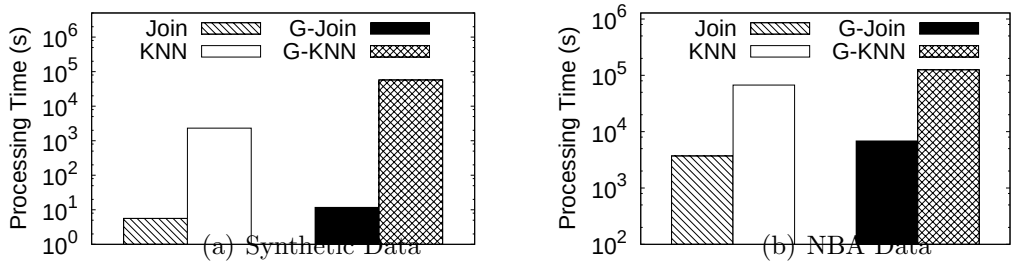
where the default values are in **bold** font.

dimensionality d	2, **3**, 4, 5
number of objects n	**5k**, 7.5k, 10k, 12.5k, 15k
edge length h	**0.02**, 0.04, 0.06, 0.08, 0.10
number of instances m	100, **200**, 400, 600, 800
K	5, **10**, 15, 20, 25
ϕ	0.1, 0.3, **0.5**, 0.7, 0.9
object location	uniform, normal, **anti-correlated**
weight distribution	uniform, **normal**

Table 6.2: Parameter settings

6.5.1 Overall Performance

Figure 6.4 reports the results of the evaluation on processing time on **Join**, **G-Join** and their corresponding baseline algorithms **KNN**, **G-KNN** on both synthetic and real data. As shown, **Join** and **G-Join** are up to 3 orders of magnitude more efficient than their baseline versions on synthetic data. The improvement is less significant on NBA data because in NBA dataset, objects' MBRs largely overlap so that it is very hard to prune an object. In the rest of the experiments we no longer evaluate the baseline algorithms since their performance is much worse than the techniques proposed in this chapter.



(a) Synthetic Data (b) NBA Data

Figure 6.4: Compare with Baseline Algorithms

6.5.2 Accuracy

We evaluate the accuracy of **G-Join** in this part. As the ϕ-quantile group-base similarity join is NP-hard and no efficient algorithm exists, we produce the exact solution using exhaustive search which is exponential to the number of instances and very slow. So we conduct a very small scale experiment as follows. Each dataset contains 500 multi-valued objects and each object consists of 4 instances. Other parameters use the default setting in Table 6.2.

To evaluate the accuracy of **G-Join**, we use two error metrics. The first is the average distance error ratio. For $1 \leq i \leq K$, $approx(i)$ denotes the group-base ϕ-quantile distance of the top-i-th object pair output by **G-Join**, and $exact(i)$ denotes the group-base distance of the top-i-th object pair in the exact solution.

$$err\_ratio = \frac{\sum_{i=1}^{K} \frac{|approx(i) - exact(i)|}{exact(i)}}{K}$$

The second measure records the "missed" object pairs, namely the object pairs that are missed in the approximate results output by **G-Join**. Let $approx$ denote the K object pairs output by **G-Join**, $exact$ denotes the K object pairs in the exact solution.

$$miss\_ratio = 1 - \frac{|approx \cap exact|}{K}$$

We report the results in Table 6.3 where the object distribution varies, and in Table 6.4 where the distribution of weights varies. Both demonstrate that **G-Join** is highly accurate and more accurate than the theoretical guarantee in Theorem 6.5.

6.5.3 Evaluating Effects by Different Settings

We study the scalability of our algorithms regarding different ϕ values, number of objects, number of instances ($\mathcal{M}$), lengths of MBR edges (h), K, and the dimen-

	err_ratio	miss_ratio
anti	0.037	0.03
unif	0.029	0.02
norm	0.036	0.02

Table 6.3: Vary Objects Distribution

	err_ratio	miss_ratio
unif	0	0
norm	0.037	0.03

Table 6.4: Vary Weight Distribution

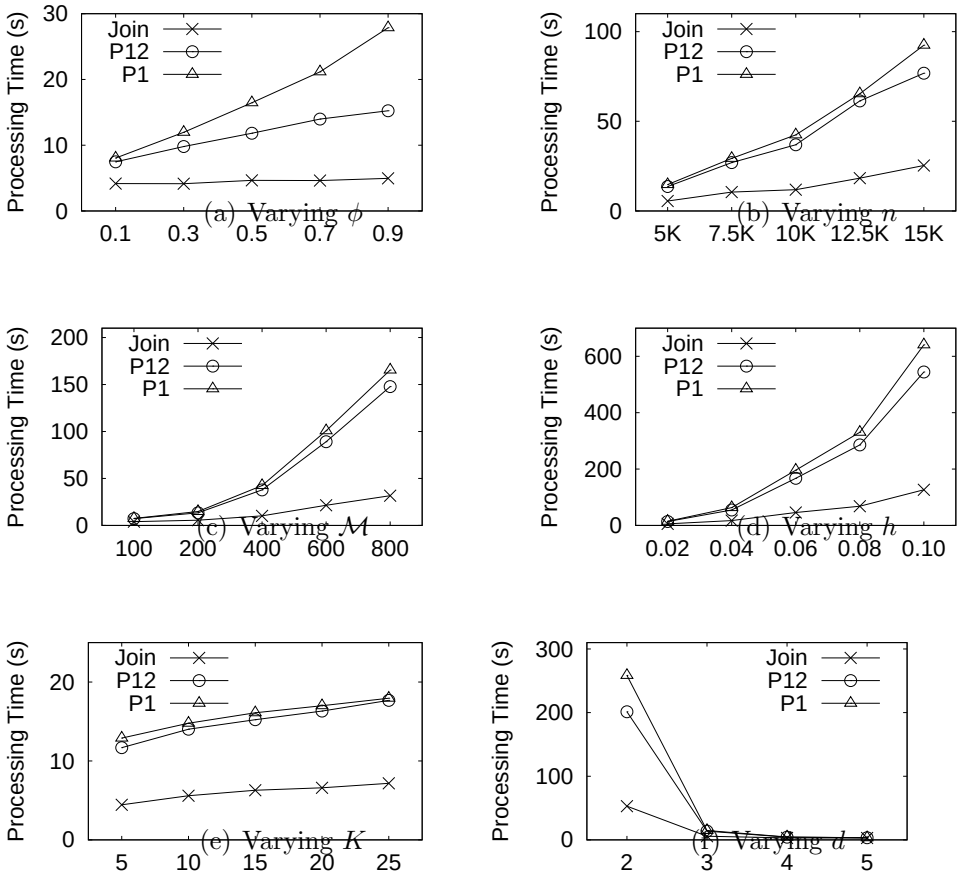


Figure 6.5: Scalability of **Join**

sionality d.

Figure 6.5 reports the scalability of **Join** regarding various parameters. It shows that **Join** is not very sensitive to different ϕ and K values, while quite sensitive to other parameters. With a larger number of objects and instances, more objects and instances are involved in the computation, thus incurring higher computation cost. With large h values, the MBRs of objects are more likely to overlap with each

other and hence the pruning power is impaired. With the increase of dimensionality d, when the MBR edge length is fixed, the average area of MBRs gets smaller compared to the whole data space; consequently, the power of the pruning rules becomes more significant. The comparison with **P1** and **P12** illustrates the efficiency of the Pruning Rule 2 and Pruning Rule 3 developed in **Join**. Note that we do not evaluate the performance of **Join** after all three pruning rules are excluded, since Pruning Rule 1 is simple yet very effective. After removing Pruning Rule 1 **Join** algorithm fails to terminate in a reasonable time.

Figure 6.6 reports the scalability of **G-Join** regarding various parameters compared with **P4** in which only Pruning Rule 4 is applied. As illustrated in the figure, **G-Join** substantially outperforms **P4** in all parameter settings, leading to the conclusion that Pruning Rule 5 is very efficient in practice. The trends observed are similar to those in Figure 6.5. **G-Join** is not very sensitive to different ϕ and K values. The performance of **G-Join** degrades with larger number of objects and number of instances since more instances are involved in the computations. With a larger average MBR length, **G-Join** requires more computation time since the higher degree of overlapping leads to weaker pruning ability. **G-Join** performs better when the dimensionality increases because once the MBR edge length is fixed, in higher dimensions the average area of an MBR gets smaller so that an object has a larger to be pruned.

6.6 Extensions

Techniques proposed in this chapter can also be applied to other variations of similarity joins over multi-valued objects. In this section, we define several variations and briefly discuss the techniques.

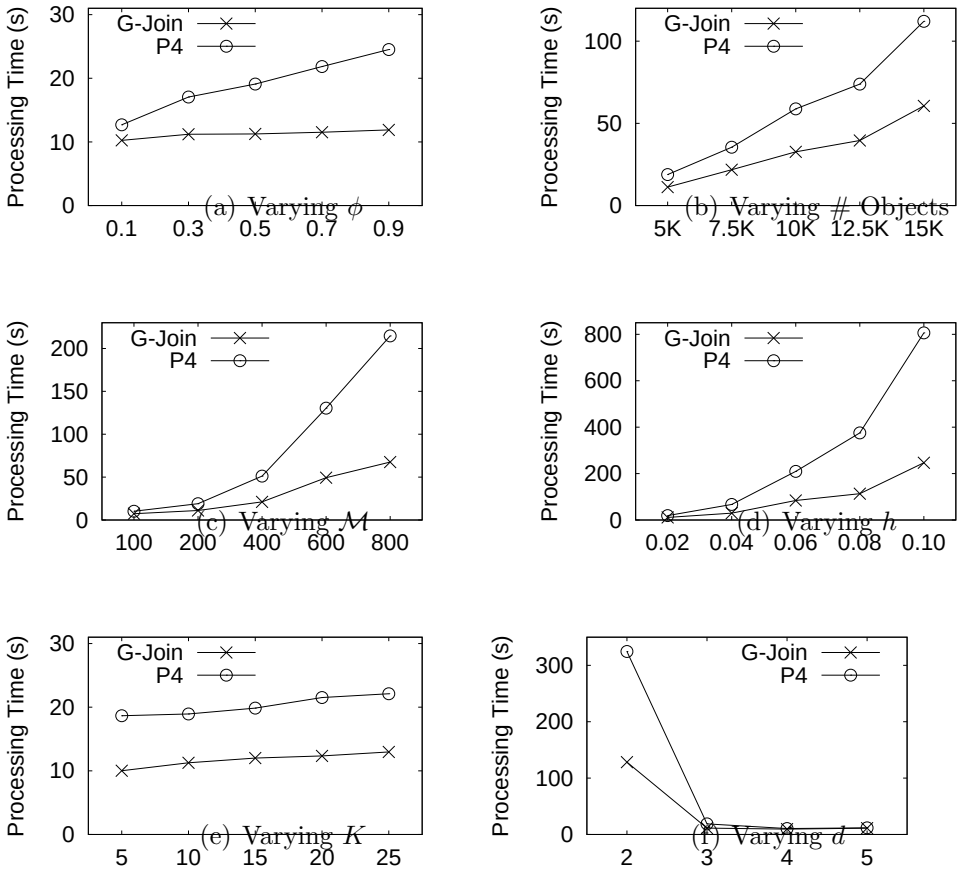


Figure 6.6: Scalability of **G-Join**

Distance Threshold based Similarity Join on Multi-Valued Objects

Top k and threshold based approaches are two typical mechanism to select a limited number of results which are of most interest to the users. In distance threshold based similarity join on multi-valued objects, a distance threshold γ is pre-given by users according to their preference and domain knowledge, only object pairs with quantile distance not larger than γ will be retrieved.

Problem Definition. Given a $\phi \in (0, 1]$, a distance threshold γ, two sets of multi-valued objects $\mathcal{U}$ and $\mathcal{V}$ in the d-dimensional space, a threshold based similarity join retrieves pairs of objects from $\mathcal{U} \times \mathcal{V}$ with quantile distance not over γ, namely, $\{(U, V)|U \in \mathcal{U}, V \in \mathcal{V}, dist(U, V) \leq \gamma\}$. Here the distance metrics $dist$ could be

either ϕ-quantile distance or ϕ-quantile group base distance.

In Algorithm 13, the Seeding phase is no longer required since a distance threshold is pre-given. Similarity join processing algorithms in Section 4.2 and 5.2 can be directly applied to solving distance threshold based similarity join on multi-valued objects.

Multi-way Distance Threshold based Similarity Join on Multi-Valued Objects

We study pairwise similarity join in this chapter where two multi-valued objects sets are combined together to retrieve the most similar object pairs. Multiway spatial joins, on the other hand, involve an arbitrary number of datasets where the join condition is specified over any two datasets. Next we discuss multi-way distance threshold based similarity join on multi-valued objects.

Problem Definition. Given n multi-valued datasets $\mathcal{U}_1$, ..., $\mathcal{U}_n$, a distance threshold γ_{ij} is specified between two datasets $\mathcal{U}_i$ and $\mathcal{U}_j$, retrieve all n-objects that respects the distance thresholds, $\{(U_1, ..., U_n) | 1 \leq k \leq n, U_k \in \mathcal{U}_k \ \& \ \forall ij, dist(U_i, U_j) \leq \gamma_{ij}\}$. Here the distance threshold *dist* could be either ϕ-quantile distance or ϕ-quantile group-base distance.

Note that the distance constraint could be applied to any pair of datasets. [MP01] studies multi-way join on spatial data where a *synchronous traversal* paradigm is applied to process the indexes (e.g., R-trees) of all joined datasets. Our filtering techniques could be directly plugged into the framework in [MP01] to facilitate pruning based on the given pairwise distance constraints.

Bichromatic and homochromatic Top k Similarity Join on Multi-Valued Objects

Consider that each object is assigned either blue or red color. A chromatic query adds an additional constraint compared to its nonchromatic version; that

is, only the results that meet the color requirement are considered. A bichromatic (homochromatic) top k similarity join retrieves k most similar object pairs with different (the same) colors.

A straightforward solution for bichromatic and homochromatic top k similarity join on multi-valued objects based on ϕ-quantile distance or ϕ-quantile group-base distance is to construct two indexes for each color for each dataset involved. Consider the problem definitions in 6.1.1, for the dataset $\mathcal{U}$ we build two R-tree based indexes, $R_{\mathcal{U},red}$ and $R_{\mathcal{U},blue}$, for the color red and blue respectively. Similarly, we build $R_{\mathcal{V},red}$ and $R_{\mathcal{V},blue}$ for dataset $\mathcal{V}$. Bichromatic top k similarity join will be conducted between the indexes with different colors, i.e., $R_{\mathcal{U},red} \times R_{\mathcal{V},blue}$ and $R_{\mathcal{U},blue} \times R_{\mathcal{V},red}$, while homochromatic top k similarity join will be conducted between the indexes with the same color. The pruning techniques developed in our chapter could be plugged in to speed up the query processing.

6.7 Conclusion

We study the problem of top k similarity join over multi-valued objects. The distance/similarity between two multi-valued objects is measured using quantile based distance to capture the relative instance distribution. A filtering-refinement framework is developed, along with novel, efficient and effective distance, statistic and weight based pruning techniques. Comprehensive experimental study over both real and synthetic datasets demonstrates the efficiency and scalability of our techniques.

Chapter 7

Conclusion and Future Work

This chapter concludes the thesis by first summarizing the major contributions in Section 7.1. Then, Section 7.2 presents several possible directions of future work on developing advanced querying and indexing techniques on uncertain data.

7.1 Conclusion

Due to a variety of reasons including data randomness and incompleteness, noise, privacy, etc., uncertainty is inherent in many important applications, such as location-based services (LBS), sensor network monitoring and radio-frequency identification (RFID). Recently, considerable research efforts have been devoted into the field of uncertainty-aware spatial query processing such that the uncertainty of the data can be effectively and efficiently tackled. Numerous query types have been re-investigated under the uncertain semantics and many database management systems have been developed especially for uncertain data, as we summarized in Chapter 2.

In this thesis, we have studied fundamental problems regarding advanced spatial query processing and indexing techniques on uncertain data. In particular, the

following four problems have been identified and tackled:

(1) Find Top k Influential Facilities.

(2) Identify Top k Dominating Objects.

(3) Range Search on Uncertain Trajectories.

(4) Top k Similarity Join

Our main contributions are stated as follows:

Find Top k Influential Facilities

Based on a new ranking semantics, we propose a new model to evaluate the influences of the facilities over a set of uncertain objects and develop effective and efficient algorithms by utilizing two uncertain objects indexing techniques, R-tree and U-Quadtree respectively. A set of pruning techniques are proposed to significantly improve the performance of the *filtering* and *refinement* algorithms by reducing the number of uncertain objects and facilities in the computation. We further develop efficient randomized algorithms with accuracy guarantee to tackle uncertain objects with massive number of instances. Comprehensive experiments convincingly demonstrate the effectiveness and efficiency of our techniques.

Identify Top k Dominating Objects

We formally introduce a top k dominating query for multi-dimensional uncertain objects based on the *parameterized ranking* semantics, which is important in the multi-criteria decision analysis when users cannot explicitly provide a proper scoring function. We formally define a new model for the top k dominating query on multi-dimensional uncertain data. By utilizing the popular R-tree indexing techniques as well as spatial based and rank score based pruning techniques, we develop

an effective and efficient algorithm following the *filtering and verification* paradigm. We further improve the performance of the algorithm based on some simple statistics information of the objects. Our experiments convincingly demonstrate the effectiveness and efficiency of our techniques.

Range Search on Uncertain Trajectories

To tame the uncertainty of trajectory data collected in a wide spectrum of applications, we formally define the problem of range search on uncertain trajectories and propose effective filtering and query processing techniques to support range search on uncertain trajectories which are modeled by Markov Chains. Then we introduce an indexing structure where the summaries of the segments are organized by an SS-Tree, as well as a general framework to support range search on uncertain trajectories following the filtering and refinement paradigm. To enhance the filtering capabilities, a partition based filtering technique is developed to further enhance the filtering capabilities. Comprehensive experiments on real-life and synthetic datasets demonstrate the effectiveness and efficiency of our techniques.

Top k Similarity Join

We formalize the problem of top k similarity join over multi-valued objects, regarding two types of quantile-based distance metrics. Then, we propose efficient and effective algorithms to compute the top k similarity join results. Particularly, we propose novel and efficient distance, statistic and weight based pruning techniques to significantly speed up the computation. Comprehensive experiments are conducted on both real and synthetic data to demonstrate the efficiency and effectiveness of our techniques.

Our main contributions can be summarized by a set of effective filtering and

verification techniques as well as efficient searching algorithms on spatial uncertain data, and extensive performance evaluations for verifying the efficiency and effectiveness of our proposed techniques on corresponding problems.

7.2 Future Work

Observing the increasing popularity of spatial query processing on uncertain date in modern applications, a number of related problems that are worth further study have also come to our attention. We list three interesting problems to be investigated in the future.

7.2.1 Spatial Query on Uncertain Data with Complex Correlations

Currently, most existing query approaches are based on the assumption of simple correlations among uncertain data such as independence or mutual exclusiveness. While this independence assumption holds in some real applications, in other scenarios, however, application data may exhibit dependencies and correlate with each other. For example, we put several sensors in a warehouse to monitor humidity and temperature. If some sensors are spatially close to each other in a small monitoring region, the collection data of them may tend to be similar and appear correlated with each other. Therefore, a possible direction of future work is to manipulate complex correlations of uncertain data in probabilistic spatial queries. There are some key challenges. Firstly, novel model is required to derive correlations from real environment. Secondly, efficient and effective techniques need to be developed to process various queries on probabilistic data given the presence of complex correlations. Thirdly, efficient and effective test techniques are required to evaluate

the accuracy of query results.

7.2.2 Top k Dominating Query on Uncertain Objects with Massive Number of Instances

In some scenarios, each object may contain a large number of instances. For example, we want to evaluate the performance of NBA player based on five years statistic data. The techniques proposed in Chapter 4 could not apply to the case, due to the limitation of computing equipments and *parameterized ranking* semantics, where the generation functions would give negative results when we perform a large number of multiplications. Furthermore, the simple sample methods are not enough to guarantee the accuracy of dominating query. Therefore, it is challenge to propose efficient and effective techniques to processing top k dominating query on uncertain objects with massive number of instances.

7.2.3 Popular Sub-route Suggestions on Uncertain Trajectories

To save the energy and the communication cost, a taxi may report its location at a low frequency. The time period between two check-in positions might be long in Geo-social applications such as bike routes and tourist routes. Consequently, a large volume of spatio-temporal data with low sampling rate is described by *uncertain trajectories* where the possible locations of a moving object between two subsequent observations are captured by a time-dependent random variable (i.e., a stochastic process). When a tourist want to visit a landscape near his/her current position, we can use the collected history data to give route suggestions. In the history data, the route between the two point maybe uncertain. Therefore, it is

challenge to propose efficient and effective techniques to mine the history route data to give the tourist some route suggestions with ranking.

Bibliography

[ACTY09] Pankaj K. Agarwal, Siu-Wing Cheng, Yufei Tao, and Ke Yi. Indexing uncertain data. In *PODS*, pages 137–146, 2009.

[AF12] Fabrizio Angiulli and Fabio Fassetti. Indexing uncertain data in general metric spaces. *IEEE Trans. Knowl. Data Eng.*, 24(9):1640–1657, 2012.

[AKG87] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. In *SIGMOD Conference*, pages 34–48, 1987.

[AKG91] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78(1):158–187, 1991.

[AY08] Charu C. Aggarwal and Philip S. Yu. On high dimensional indexing of uncertain data. In *ICDE*, pages 1460–1461, 2008.

[BDJ+07] Douglas Burdick, Prasad M. Deshpande, T. S. Jayram, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. Olap over uncertain and imprecise data. *VLDB J.*, 16(1):123–144, 2007.

[BEK+11] Thomas Bernecker, Tobias Emrich, Hans-Peter Kriegel, Matthias

Renz, Stefan Zankl, and Andreas Züfle. Efficient probabilistic reverse nearest neighbor query processing on uncertain data. *PVLDB*, 4(10):669–680, 2011.

[BGK+07] Christian Böhm, Michael Gruber, Peter Kunath, Alexey Pryakhin, and Matthias Schubert. Prover: Probabilistic video retrieval using the gauss-tree. In *ICDE*, pages 1521–1522, 2007.

[BKS93] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient processing of spatial joins using r-trees. In *SIGMOD Conference*, pages 237–246, 1993.

[BKS01] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[BPS06] Christian Böhm, Alexey Pryakhin, and Matthias Schubert. Probabilistic ranking queries on gaussians. In *SSDBM*, pages 169–178, 2006.

[Bur] U.S. Census Bureau. `http://www.census.gov/geo/www/tiger`.

[CKP03] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD Conference*, pages 551–562, 2003.

[CKP04] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Querying imprecise data in moving object environments. *IEEE Trans. Knowl. Data Eng.*, 16(9):1112–1127, 2004.

[CLRS01] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction To Algorithms. Chapter 9: Medians and Order Statistics*. MIT Press, 2001.

[CLW+10] Muhammad Aamir Cheema, Xuemin Lin, Wei Wang, Wenjie Zhang, and Jian Pei. Probabilistic reverse nearest neighbor queries on uncertain data. *IEEE Trans. Knowl. Data Eng.*, 22(4):550–564, 2010.

[CLY09] Graham Cormode, Feifei Li, and Ke Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, pages 305–316, 2009.

[CMTV00] Antonio Corral, Yannis Manolopoulos, Yannis Theodoridis, and Michael Vassilakopoulos. Closest pair queries in spatial databases. In *SIGMOD Conference*, pages 189–200, 2000.

[CP07] Yun Chen and Jignesh M. Patel. Efficient evaluation of all-nearest-neighbor queries. In *ICDE*, pages 1056–1065, 2007.

[CPK03] Reynold Cheng, Sunil Prabhakar, and Dmitri V. Kalashnikov. Querying imprecise data in moving object environments. In *ICDE*, pages 723–725, 2003.

[CSP+06] Reynold Cheng, Sarvjeet Singh, Sunil Prabhakar, Rahul Shah, Jeffrey Scott Vitter, and Yuni Xia. Efficient join processing over uncertain data. In *CIKM*, pages 738–747, 2006.

[CXP+04] Reynold Cheng, Yuni Xia, Sunil Prabhakar, Rahul Shah, and Jeffrey Scott Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *VLDB*, pages 876–887, 2004.

[DS07] Nilesh N. Dalvi and Dan Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.

[EKM+12a] Tobias Emrich, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, and Andreas Züfle. Indexing uncertain spatio-temporal data. In *CIKM*, pages 395–404, 2012.

[EKM+12b] Tobias Emrich, Hans-Peter Kriegel, Nikos Mamoulis, Matthias Renz, and Andreas Züfle. Querying uncertain spatio-temporal data. In *ICDE*, pages 354–365, 2012.

[EN11] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, 2011.

[Fal88] Christos Faloutsos. Gray codes for partial match and range queries. pages 1381–1393, 1988.

[FB74] Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 1974.

[FZGZ13] Xing Feng, Xiang Zhao, Yunjun Gao, and Ying Zhang. Probabilistic top-k dominating query over sliding windows. In *APWeb*, pages 782–793, 2013.

[GJ00] Michael M. Güntzer and Dieter Jungnickel. Approximate minimization algorithms for the 0/1 knapsack and subset-sum problem. *Oper. Res. Lett.*, 26(2):55–66, 2000.

[GUP06] J. Galindo, A. Urrutia, and M. Piattini. *Fuzzy databases: modeling, design, and implementation*. Idea Group Pub., 2006.

[Gut84] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*, pages 47–57, 1984.

[GZM09] Tingjian Ge, Stanley Zdonik, and Samuel Madden. Top-k queries on uncertain data: On score distribution and typical answers. In *SIGMOD*, 2009.

[HJR97] Yun-Wu Huang, Ning Jing, and Elke A. Rundensteiner. Spatial joins using r-trees: Breadth-first traversal with global optimizations. In *VLDB*, pages 396–405, 1997.

[HKL+09] Wook-Shin Han, Jaehwa Kim, Byung Suk Lee, Yufei Tao, Ralf Rantzau, and Volker Markl. Cost-based predictive spatiotemporal join. *IEEE Trans. Knowl. Data Eng.*, 21(2):220–233, 2009.

[HLP88] G. H. Hardy, J. E. Littlewood, and G. Plya. *Inequalities*. Cambridge University Press, 1988.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[HPZL08a] Ming Hua, Jian Pei, Wenjie Zhang, and Xuemin Lin. Efficiently answering probabilistic threshold top-k queries on uncertain data. In *ICDE*, pages 1403–1405, 2008.

[HPZL08b] Ming Hua, Jian Pei, Wenjie Zhang, and Xuemin Lin. Ranking queries on uncertain data: a probabilistic threshold approach. In *SIGMOD Conference*, pages 673–686, 2008.

[HS98] Gísli R. Hjaltason and Hanan Samet. Incremental distance join algorithms for spatial databases. In *SIGMOD Conference*, pages 237–248, 1998.

[HS02] Gísli R. Hjaltason and Hanan Samet. Speeding up construction of pmr quadtree-based spatial indexes. *VLDB J.*, 11(2):109–137, 2002.

[HWQ+11] Jin Huang, Zeyi Wen, Jianzhong Qi, Rui Zhang, Jian Chen, and Zhen He. Top-k most influential locations selection. In *CIKM*, pages 2377–2380, 2011.

[IJ84] Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[JKM+98] H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, and Torsten Suel. Optimal histograms with quality guarantees. In *VLDB*, pages 275–286, 1998.

[JLSY14] Hoyoung Jeung, Hua Lu, Saket Sathe, and Man Lung Yiu. Managing evolving uncertainty in trajectory databases. *IEEE Trans. Knowl. Data Eng.*, 26(7):1692–1705, 2014.

[JXW+08] Ravi Jampani, Fei Xu, Mingxi Wu, Luis Leopoldo Perez, Christopher M. Jermaine, and Peter J. Haas. Mcdb: a monte carlo approach to managing uncertain data. In *SIGMOD*, 2008.

[KKPR06] Hans-Peter Kriegel, Peter Kunath, Martin Pfeifle, and Matthias Renz. Probabilistic similarity join on uncertain data. In *DASFAA*, pages 295–309, 2006.

[KKR07] Hans-Peter Kriegel, Peter Kunath, and Matthias Renz. Probabilistic nearest-neighbor query on uncertain objects. In *DASFAA*, pages 337–348, 2007.

[KM00] Flip Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *SIGMOD*, pages 201–212, 2000.

[KMZ10] Hideaki Kimura, Samuel Madden, and Stanley B. Zdonik. Upi: A primary index for uncertain databases. *PVLDB*, 3(1):630–637, 2010.

[KN96] Edwin M. Knorr and Raymond T. Ng. Finding aggregate proximity relationships and commonalities in spatial data mining. *IEEE Trans. Knowl. Data Eng.*, 8(6):884–897, 1996.

[KT75] S. Karlin and H.M. Taylor. *A First Course in Stochastic Processes*. Academic Press, 1975.

[LC09a] Xiang Lian and Lei Chen. Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data. *VLDB J.*, 18(3):787–808, 2009.

[LC09b] Xiang Lian and Lei Chen. Top-k dominating queries in uncertain databases. In *EDBT*, pages 660–671, 2009.

[LC10] Xiang Lian and Lei Chen. A generic framework for handling uncertain data with local correlations. *PVLDB*, 4(1):12–21, 2010.

[LC13] Xiang Lian and Lei Chen. Probabilistic top-k dominating queries in uncertain databases. *Inf. Sci.*, 226:23–46, 2013.

[Lee92] Suk Kyoon Lee. Imprecise and uncertain information in databases: An evidential approach. In *ICDE*, pages 614–621, 1992.

[LS08] Vebjorn Ljosa and Ambuj K. Singh. Top-k spatial joins of probabilistic objects. In *ICDE*, pages 566–575, 2008.

[LSD09] Jian Li, Barna Saha, and Amol Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1), 2009.

[LSD11] Jian Li, Barna Saha, and Amol Deshpande. A unified approach to ranking in probabilistic databases. *VLDB J.*, 20(2):249–275, 2011.

[LSS96] Ee-Peng Lim, Jaideep Srivastava, and Shashi Shekhar. An evidential reasoning approach to attribute value conflict resolution in database integration. *IEEE Trans. Knowl. Data Eng.*, 8(5):707–723, 1996.

[LWHS06] Min-Jae Lee, Kyu-Young Whang, Wook-Shin Han, and Il-Yeol Song. Transform-space view: Performing spatial join in the transform space using original-space indexes. *IEEE Trans. Knowl. Data Eng.*, 18(2):245–260, 2006.

[LYZZ07] Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. Selecting stars: The k most representative skyline operator. In *ICDE*, pages 86–95, 2007.

[LZZC11] Xuemin Lin, Ying Zhang, Wenjie Zhang, and Muhammad Aamir Cheema. Stochastic skyline operator. In *ICDE*, pages 721–732, 2011.

[MBJ13] Katarzyna Musial, Marcin Budka, and Krzysztof Juszczyszyn. Creation and growth of online social network - how do social networks evolve? *World Wide Web*, 16(4):421–447, 2013.

[Mee03] Ronald Meester. *A Natural Introduction to Probability Theory.* Birkhäuser, 2003.

[MKM08] Yiming Ma, Dmitri V. Kalashnikov, and Sharad Mehrotra. Toward managing uncertain spatial information for situational awareness applications. *IEEE Trans. Knowl. Data Eng.*, 20(10), 2008.

[MLSC13] Chunyang Ma, Hua Lu, Lidan Shou, and Gang Chen. Ksq: Top-(k) similarity query on uncertain trajectories. *IEEE Trans. Knowl. Data Eng.*, 25(9):2049–2062, 2013.

[MP01] Nikos Mamoulis and Dimitris Papadias. Multiway spatial joins. *ACM Trans. Database Syst.*, 26(4):424–475, 2001.

[NZE+13] 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.

[PJ99] Dieter Pfoser and Christian S. Jensen. Capturing the uncertainty of moving-object representations. In *SSD*, pages 111–132, 1999.

[PJLY07] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. Probabilistic skylines on uncertain data. In *VLDB*, pages 15–26, 2007.

[PKZT01] Dimitris Papadias, Panos Kalnis, Jun Zhang, and Yufei Tao. Efficient olap operations in spatial data warehouses. In *SSTD*, pages 443–459, 2001.

[PM97] Apostolos Papadopoulos and Yannis Manolopoulos. Performance of nearest neighbor queries in r-trees. In *ICDT*, 1997.

[PTFS05] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1), 2005.

[RDS07] Christopher Re, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, pages 886–895, 2007.

[RSV01] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases: With Application to GIS*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2001.

[SAS06] Jagan Sankaranarayanan, Houman Alborzi, and Hanan Samet. Distance join queries on spatial networks. In *GIS*, pages 211–218, 2006.

[SBHW06] Anish Das Sarma, Omar Benjelloun, Alon Y. Halevy, and Jennifer Widom. Working models for uncertain data. In *ICDE*, page 7, 2006.

[SCL+12] Zhitao Shen, Muhammad Aamir Cheema, Xuemin Lin, Wenjie Zhang, and Haixun Wang. Efficiently monitoring top-k pairs over sliding windows. In *ICDE*, pages 798–809, 2012.

[SIC07] Mohamed A. Soliman, Ihab F. Ilyas, and Kevin Chen-Chuan Chang. Top-k query processing in uncertain databases. In *ICDE*, pages 896–905, 2007.

[SMP+07] Sarvjeet Singh, Chris Mayfield, Sunil Prabhakar, Rahul Shah, and Susanne E. Hambrusch. Indexing uncertain categorical data. In *ICDE*, pages 616–625, 2007.

[SS94] Moshe Shaked and J.G. Shanthinkumar. *Stochastic Orders and Their Applications*. Probability and mathematical statistics. Academic Press, 1994.

[SS06] Mehdi Sharifzadeh and Cyrus Shahabi. The spatial skyline queries. In *VLDB*, pages 751–762, 2006.

[TCX+05] Yufei Tao, Reynold Cheng, Xiaokui Xiao, Wang Kay Ngai, Ben Kao, and Sunil Prabhakar. Indexing multi-dimensional uncertain data with

arbitrary probability density functions. In *VLDB*, pages 922–933, 2005.

[The03] Yannis Theodoridis. The r-tree-portal. `http://www.rtreeportal.org`, 2003.

[TPM11] Eleftherios Tiakas, Apostolos N. Papadopoulos, and Yannis Manolopoulos. Progressive processing of subspace dominating queries. *VLDB J.*, 20(6):921–948, 2011.

[TWHC04] Goce Trajcevski, Ouri Wolfson, Klaus Hinrichs, and Sam Chamberlain. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.*, 29(3):463–507, 2004.

[TXC07] Yufei Tao, Xiaokui Xiao, and Reynold Cheng. Range search on multidimensional uncertain data. *ACM Trans. Database Syst.*, 32(3):15, 2007.

[WÖF+11] Raymond Chi-Wing Wong, M. Tamer Özsu, Ada Wai-Chee Fu, Philip S. Yu, Lian Liu, and Yubao Liu. Maximizing bichromatic reverse nearest neighbor for lp -norm in two- and three-dimensional spaces. *VLDB J.*, 20(6), 2011.

[WQWZ09] Fang Wei, Weining Qian, Chen Wang, and Aoying Zhou. Detecting overlapping community structures in networks. *World Wide Web*, 12(2):235–261, 2009.

[XGC+13] Chuanfei Xu, Yu Gu, Lei Chen, Jianzhong Qiao, and Ge Yu. Interval reverse nearest neighbor queries on uncertain data with markov correlations. In *ICDE*, pages 170–181, 2013.

[XYCL13] Xike Xie, Man Lung Yiu, Reynold Cheng, and Hua Lu. Scalable evaluation of trajectory queries over imprecise location data. *IEEE Trans. Knowl. Data Eng.*, Accppted in 2013.

[XZKD05] Tian Xia, Donghui Zhang, Evangelos Kanoulas, and Yang Du. On computing top-t most influential spatial sites. In *VLDB*, pages 946–957, 2005.

[YLKS08] Ke Yi, Feifei Li, George Kollios, and Divesh Srivastava. Efficient processing of top-k queries in uncertain databases with x-relations. *IEEE Trans. Knowl. Data Eng.*, 20(12):1669–1682, 2008.

[YM07] Man Lung Yiu and Nikos Mamoulis. Efficient processing of top-k dominating queries on multi-dimensional data. In *VLDB*, pages 483–494, 2007.

[YM09] Man Lung Yiu and Nikos Mamoulis. Multi-dimensional top-k dominating queries. *VLDB J.*, 18(3):695–718, 2009.

[YMT06] Man Lung Yiu, Nikos Mamoulis, and Yufei Tao. Efficient quantile retrieval on multi-dimensional data. In *EDBT*, pages 167–185, 2006.

[YWN11] Da Yan, Raymond Chi-Wing Wong, and Wilfred Ng. Efficient methods for finding influential locations with adaptive grids. In *CIKM*, pages 1475–1484, 2011.

[YZXS11] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *KDD*, pages 316–324, 2011.

[ZC08] Xi Zhang and Jan Chomicki. On the semantics and evaluation of top-k queries in probabilistic databases. In *ICDE Workshops*, pages 556–563, 2008.

[ZCJ+09] Meihui Zhang, Su Chen, Christian S. Jensen, Beng Chin Ooi, and Zhenjie Zhang. Effectively indexing uncertain moving objects for predictive queries. *PVLDB*, 2(1):1198–1209, 2009.

[ZHZZ12] Kai Zheng, Zi Huang, Aoying Zhou, and Xiaofang Zhou. Discovering the most influential sites over uncertain data: A rank-based approach. *IEEE Trans. Knowl. Data Eng.*, 24(12):2156–2169, 2012.

[ZLC+10] Wenjie Zhang, Xuemin Lin, Muhammad Aamir Cheema, Ying Zhang, and Wei Wang. Quantile-based knn over multi-valued objects. In *ICDE*, 2010.

[ZLRB08] Rui Zhang, Dan Lin, Kotagiri Ramamohanarao, and Elisa Bertino. Continuous intersection joins over moving objects. In *ICDE*, pages 863–872, 2008.

[ZLZ+10a] Wenjie Zhang, Xuemin Lin, Ying Zhang, Jian Pei, and Wei Wang. Threshold-based probabilistic top-k dominating queries. *VLDB J.*, 19(2):283–305, 2010.

[ZLZ+10b] Ying Zhang, Xuemin Lin, Wenjie Zhang, Jianmin Wang, and Qianlu Lin. Effectively indexing the uncertain space. *IEEE Trans. Knowl. Data Eng.*, 22(9):1247–1261, 2010.

[ZLZ+10c] Ying Zhang, Xuemin Lin, Gaoping Zhu, Wenjie Zhang, and Qianlu Lin. Efficient rank based knn query processing over uncertain data. In *ICDE*, 2010.

[ZTZS11] Kai Zheng, Goce Trajcevski, Xiaofang Zhou, and Peter Scheuermann. Probabilistic range queries for uncertain trajectories on road networks. In *EDBT*, pages 283–294, 2011.

[ZXL+12] Wenjie Zhang, Jing Xu, Xin Liang, Ying Zhang, and Xuemin Lin. Top-k similarity join over multi-valued objects. In *DASFAA (1)*, pages 509–525, 2012.

[ZZL+14] Ying Zhang, Wenjie Zhang, Qianlu Lin, Xuemin Lin, and Heng Tao Shen. Effectively indexing the multidimensional uncertain objects. *IEEE Trans. Knowl. Data Eng.*, 26(3):608–622, 2014.

[ZZLL12] Ying Zhang, Wenjie Zhang, Qianlu Lin, and Xuemin Lin. Effectively indexing the multi-dimensional uncertain objects for range searching. In *EDBT*, pages 504–515, 2012.

[ZZXZ12] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *ICDE*, pages 1144–1155, 2012.