Finding Top k Most Influential Spatial Facilities over Uncertain Objects

Liming Zhan Ying Zhang Wenjie Zhang Xuemin Lin

School of Computer Science & Engineering
The University of New South Wales, Sydney NSW 2052, Australia
{zhanl, yingz, zhangw, lxue}@cse.unsw.edu.au

ABSTRACT

Uncertainty is inherent in many important applications, such as location-based services (LBS), sensor monitoring and radio-frequency identification (RFID). Recently, considerable research efforts have been put into the field of uncertainty-aware spatial query processing. In this paper, we study the problem of finding top k most influential facilities over a set of uncertain objects, which is an important 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. Comprehensive experiments on real datasets demonstrate the effectiveness and efficiency of our techniques.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms, Performance

Keywords

Spatial, uncertain

1. INTRODUCTION

Bichromatic reverse nearest neighbor (BRNN) query has been extensively studied as an important spatial operator ever since it was introduced in [10] 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 a set of $\mathcal{O}$ of objects (e.g., car, person), the influence of a facility F can be evaluated by the number of objects whose nearest neighbors are 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 [20] 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(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 example applications.

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(b), the item A may appear at two positions a_1 and a_2 with the same probability. The facilities in Figure 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 users possible location regarding the nearby towers. In Figure 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.

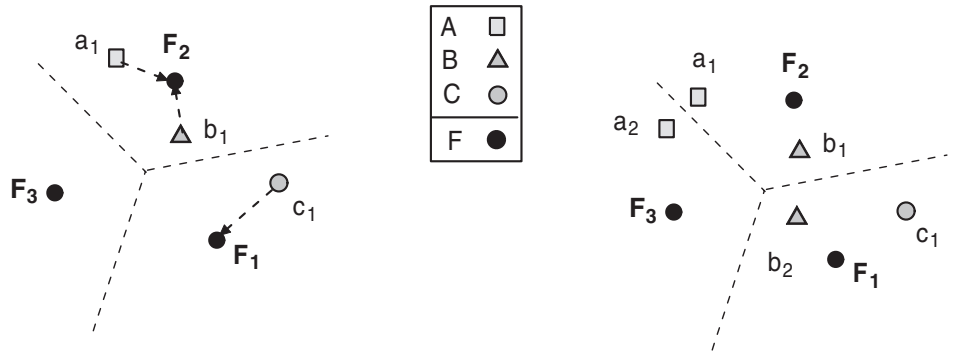
Challenges. Motivated by the above examples, it is desirable to study the problem of finding top k most influential facilities over uncertain objects. The challenges are twofold.

Firstly, unlike the traditional spatial database in which an object only contributes to the influence score of one facility[1], it is nontrivial to evaluate the influence of a facility because of the existence of multiple instances. As shown in Figure 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.

In the paper, we follow the popular *possible world semantics* (See Section 3.1 for a formal definition), and the influence of each facility is modeled as a influence score distribution. The definition

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

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.



(a) A set of ***Objects*** and Facilities

(b) A set of ***Uncertain Objects*** and Facilities

Figure 1: Motivation Example

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 [14] 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.

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 in Section 3 we show that 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. Therefore, in the paper, 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 [24], namely U-Quadtree, we further significantly improve the performance of the algorithm in terms of CPU and I/O costs.

Contributions. Our contribution can be summarized as follows.

- 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.
- Comprehensive experiments demonstrate the effectiveness and efficiency of our techniques.

Organization of the paper. The remainder of the paper is organized as follows. Section 2 presents the related work in the paper. Then, we formally define the problem of top k most influential facilities over uncertain objects, and introduce some preliminary work in Section 3. In Section 4, we propose our efficient algorithms based on R-tree and U-Quadtree respectively. Results of comprehensive performance studies are presented in Section 5. Finally, Section 6 concludes the paper.

2. RELATED WORK

In the last decade, a lot of work has been done in the field of uncertain data processing and management. Since our work focus on the top k query of influential facilities over uncertain data, we briefly introduce the existing work on uncertain data processing and influential facilities query.

2.1 Influential Facilities Computation

Bichromatic reverse nearest neighbor query is first introduced by Korn *et al.* in [10]. 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 [20]. 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, [9] 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 [21, 19] , aiming to find optimal area or location to set up a new facility such that it can attract the greatest number of facilities.

2.2 Uncertain Data Processing

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 [16] first relate top k queries with uncertain data. They define two types of important queries - U-Top$k$ and U-kRank, regarding discrete cases. The first one returns a set of k records which as a whole have the highest probability to be the top k results in all possible worlds, while the second one retrieves k ordered records where the i-th record has the highest probability of ranking in the i-th position among all possible worlds. Following, a large amount of work has been dedicated to top k queries with different semantics such as PT-k [8], Global-top k [22], expected rank top k [5] and unified top k [11] semantics.

Meanwhile, many spatial query techniques have been extended to solve uncertain problem. Nearest neighbor (NN) query over uncertain data is one of the most flourishing topic. Cheng *et al.* [4]

is 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. Zhang *et al.* [23] employ a rank based approach to process probabilistic kNN query, where k closest objects are returned according to their expected ranks. Cheema *et al.* [3] 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. [12], [1] also propose techniques to solve PRNN queries over uncertain data. Moreover, some indexing techniques are proposed for uncertain data such as R-tree [15], U-tree [17] and U-Quadtree [24].

2.3 Influential Facilities over Uncertain data

To our best knowledge, Zheng *et al.* [25] is the only existing work which 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* as the ranking function to define the order of the facilities with probabilistic influences. The 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* [5]. 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 result.

3. BACKGROUND

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

Table 1: The summary of notations.

3.1 Problem Definition

A point (instance) x referred in the paper, 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 the paper, and the techniques developed in the paper can be easily extended to other metric distances. In the paper, 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 paper, 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 the paper, 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 $\sum_{W \in \mathcal{W}} Pr(W) = 1.0$.

EXAMPLE 1. *In Figure 2(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 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 2(b).*

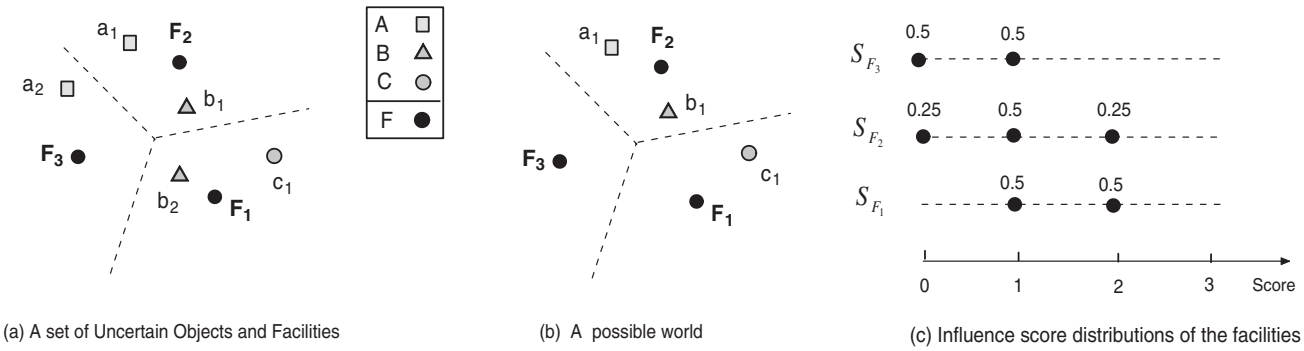
For each possible world W, let $s(F, W)$ denote the influence score of the facility F regarding W, which is the number of reverse nearest neighbors of F in W; In the paper, for each facility F, we use S_F to represent the influence score distribution of F, where $Pr(S_F = v) = \sum_{W \in \mathcal{W}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

EXAMPLE 2. *In Figure 2(b), we have $s(F_1, W) = 1$, $s(F_2, W) = 2$ and $s(F_3, W) = 0$. Figure 2(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 [14] 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 the paper, the *expected influence score* of a facility F, denoted by $I(F)$, is defined as follows.

$$I(F) = \sum_{W \in \mathcal{W}} s(F, W) \times Pr(W) \quad (1)$$

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 objects, it is cost-inhibitive to apply Equation 1 straightforwardly by enumerating all possible worlds. Therefore, we will find an alternative of Equation 1 which can be derived with reasonable computational cost.



(a) A set of Uncertain Objects and Facilities (b) A possible world (c) Influence score distributions of the facilities

Figure 2: Example for the Problem Definition

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 1 can be rewritten as follows.

$$I(F) = \sum_{W \in \mathcal{W}} (\sum_{U \in \mathcal{U}} \sigma(NN(U, W) = F)) \times Pr(W) \quad (2)$$

$$= \sum_{U \in \mathcal{U}} Pr(NN(U) = F) \quad (3)$$

$$= \sum_{U \in \mathcal{U}} \sum_{u \in U \wedge NN(u) = F} p_u \quad (4)$$

, 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 4 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 objects. Our empirical study shows that even a naive implementation of the Equation 4 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.

3.2 Expected Score vs Expected Rank

In [25], 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.

$$er(F) = \sum_{W \in \mathcal{W}} r(F, W) \times Pr(W) \quad (5)$$

, 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 [25], 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 [5], 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 the paper 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.

3.3 Preliminaries

Various indexing techniques have been proposed to organize uncertain objects. In the paper, 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 [15], 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 4 illustrates the basic idea of the R-tree based indexing approach where the MBRs of the uncertain objects are indexed by R-tree [7]. 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.

Indexing Uncertain Objects by U-Quadtree

A quadtree [6] is a space partitioning tree data structure in which a d-dimensional space is recursively subdivided into 2^d regions (cells). In [24], 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

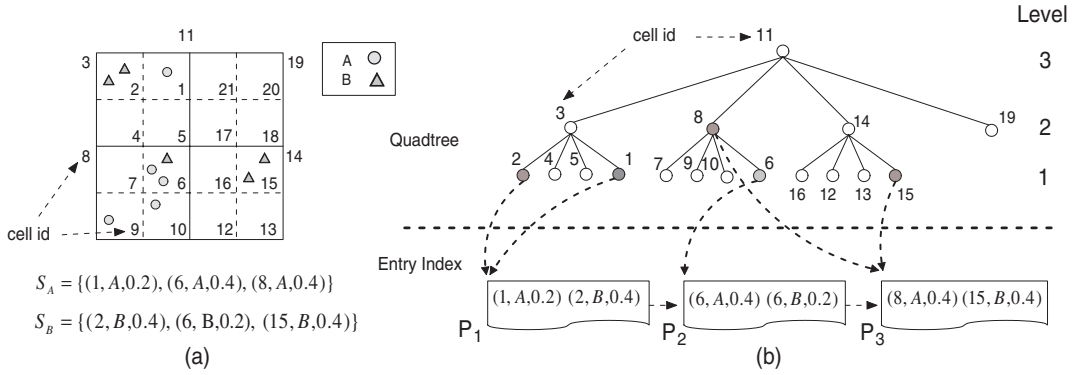


Figure 3: U**-Quadtree**

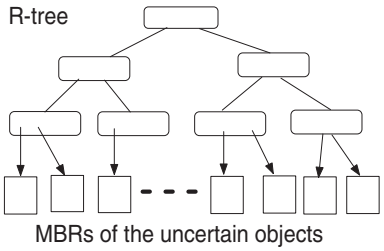


Figure 4: R**-tree based Indexing**

generated based on Hilbert curve. Figure 3 illustrates an example of the U-Quadtree.

EXAMPLE 3. *In Figure 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 [24], 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 [24].

4. APPROACH

In this section, we investigate efficient algorithms to compute the top k most influential facilities based on their *expected influence scores*. Section 4.1 presents a straightforward implementation of the algorithm. Assuming the uncertain objects are organized by R-tree, Section 4.2 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 4.3 further improves the performance of the *filtering* and the *verification* algorithms.

In the paper, 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 the paper can be easily extended to other hierarchical spatial indexing techniques.

4.1 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 4. For each instance of an uncertain object, a nearest neighbor query [13] 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

```
for each U ∈ S_U do
    for each instance u ∈ U do
        for find nearest facility F ∈ S_F do
            I(F) := I(F) + p_u;
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.

4.2 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 the paper, 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 the paper.

DEFINITION 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 [20], effective method is proposed to compute the minimal and maximal nearest neighbor distances between two rectangles. In the paper, 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 $nnd(x, \mathcal{F}) \leq nnd_{max}(R_1, R_2)$.*

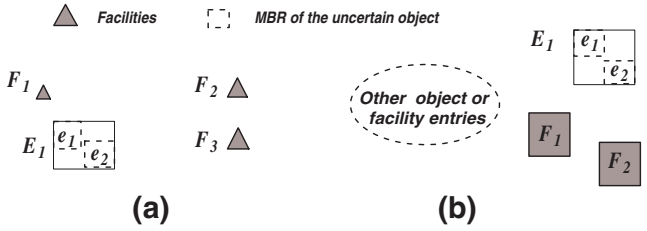


Figure 6: Motivation Example

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 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 4. *In Figure 5, 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.*

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.

In Figure 5, 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 the paper, we say a facility entry F is *disabled* if $I^+(F) < \lambda$.

Besides the facility entries, we can also *prune* object entries in the paper. In Figure 6(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 6(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 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 $\mathcal{C} := \emptyset$; $\mathcal{S} := \emptyset$; $Q := \emptyset$; $\lambda = 0$;
2 generate a new tuple T;
3 $T.e \leftarrow$ the root of $R_{\mathcal{O}}$;
4 $e_f \leftarrow$ the root of $R_{\mathcal{F}}$;
5 $T.\mathcal{F} \leftarrow e_f$; $I^-(e_f) := 0$; $I^+(e_f) :=$ # objects in $R_{\mathcal{O}}$;
6 push T into FIFO queue Q ;
7 **while** Q is not empty **do**
8 $T \leftarrow$ pop the head of Q ;
9 **if** $T.e$ is data entry and all facility entries in $T.\mathcal{F}$ are data entries **then**
10 $\mathcal{S} := \mathcal{S} \cup T$;
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 $T.\mathcal{F}$ **do**
15 **if** F is not *disabled* **then**
16 **for** each *child* entry e_f of F **do**
17 $t.\mathcal{F} := t.\mathcal{F} \cup e_f$;
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 $t.\mathcal{F}$;
20 **for** each facility entry e_f in $t.\mathcal{F}$ **do**
21 **if** $nnd\_min(t.e, e_f) > d_{max}$ **then**
22 $t.\mathcal{F} := t.\mathcal{F} - e_f$;
23 $I^+(e_f) := I^+(e_f) - agg(t.e)$;
24 **if** $I^+(e_f) < \lambda$ **then**
25 Disable F;
26 **if** $t.\mathcal{F}$ 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 $t.\mathcal{F}$ are *disabled* **then**
30 Disable t ;
31 **if** t is not *disabled* **then**
32 Push t to the tail of Q ;
33 $\mathcal{C} \leftarrow$ facilities with $I^+(F) \geq \lambda$;
34 **return** $\mathcal{S}$, $\mathcal{C}$

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 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 $T.e$ [2]. 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 $agg(t.e)$

[2]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.

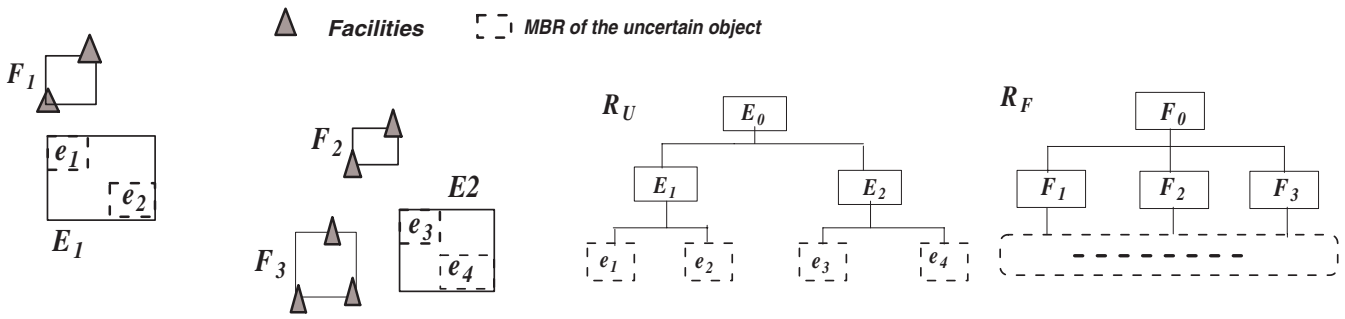


Figure 5: Motivation Example

(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 $\mathcal{S}$.

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  : C : the candidate facilities,
            S : the objects tuples,
            k
   Output : 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 the paper, 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 others 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.

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

```
   Input  : e : the R-tree entry,
            F : a set of facilities
   Output : Updated 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
```

4.3 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 [24], 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.

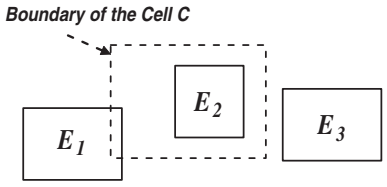


Figure 7: Containment Example

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.

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

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 7, E_2 is *fully contained* by c and E_1 is *partially contained* by c.

Algorithm 5 illustrates the details of the U-Quadtree based refinement algorithm. 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.

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

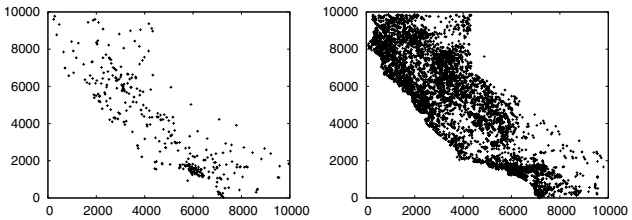
```
   Input  : C : the candidate facilities,
            S : the objects need to further explore,
            k
   Output : I : the top k influential facilities
 1 I := ∅; ;
 2 for each uncertain object U in S do
 3     for each cell c associated with U do
 4         Q := ∅; Q ← root of R_U ;
 5         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
```

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 paper. Following algorithms are evaluated.

- **Naive** The naive implementation proposed in Section 4.1.
- **RTKIS** The technique based on R-tree proposed in Section 4.2.
- **UQuadTKIS** The technique based on U-Quadtree proposed in Section 4.3.
- **UTKIS** The technique presented in [25].

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 [2]. 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 8. RT is obtained from the R-tree-Portal [18] 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.



(a) *Facilities* (b) *Objects*

Figure 8: Data distribution

All algorithms proposed in this paper 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 paper, 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.

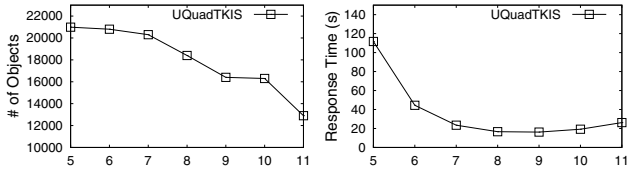
Query response time is recorded to evaluate the efficiency of the algorithms, which contains the CPU time and the I/O latency.

Table 2 lists parameters which may have an impact on our performance study. In our experiments, all parameters use default values unless otherwise specified.

Notation	Definition (Default Values)
h	height of U-Quadtree (9)
r_u	radius of uncertain object region (60)
n	number of uncertain objects ($20K$)
m	number of instances per object (200)
f	number of facility ($1K$)

Table 2: Parameter settings

5.1 Performance Tuning



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

Figure 9: Diff. U-Quadtree Height

The performance of UQuadTKIS is effected by the height (h) of the U-Quadtree. As expected, Figure 9(a) 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.

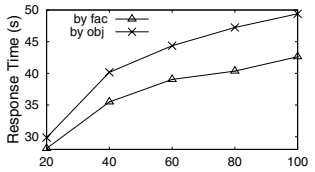


Figure 10: Access Order

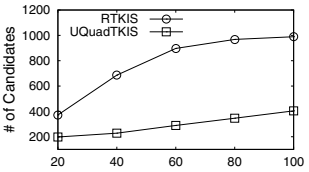


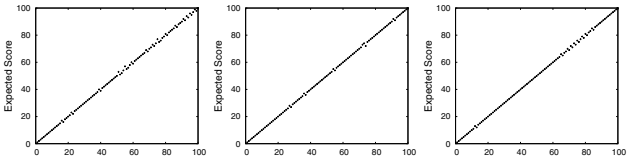
Figure 11: Candidate Size

Figure 10 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 outperform the object based strategy.

We evaluate the *filtering* effectiveness of RTKIS and UQuadTKIS in Figure 11 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.

5.2 Performance Evaluation

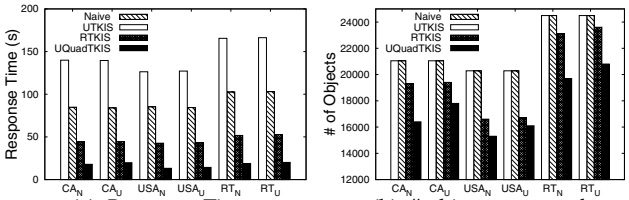
Comparing Different Ranking models. Figure 12 evaluates the similarity of different ranking models (*expected rank* and *expected score*) on three datasets CA, USA and RT respectively, which shows the scatter-plot of the ranks of 100 facilities. Particularly, each point in the scatter-plot represents a facility where x-axis and y-axis record the rank of objects based on *expected rank* model and



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

Figure 12: Result comparison

expected score model respectively. It is shown that all points line up along the diagonal, and the maximal difference of the ranks for a facility is only 2 in Figure 12 , which indicates that the results of two models are almost the same.



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

Figure 13: Impact of Data Distributions

Impact of Data Distribution. Figure 13 reports the response time and the number of uncertain objects accessed of the algorithms against different data distributions where CA_N represent 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.

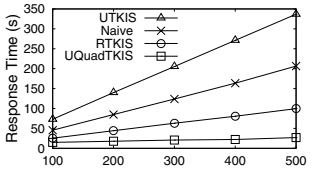


Figure 14: Diff. m

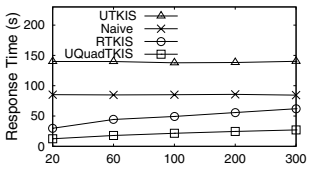


Figure 15: Diff. r_u

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 14 shows that UQuadTKIS has the best scalability against m, followed by RTKIS, Naive and UTKIS.

Impact of the radius(r_u). Figure 15 investigates the performance of four algorithms as a function of the radius size which various from 20 to 300. It is shown that the scalability of UQuadTKIS is better than that of RTKIS regarding the growth 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 10,000 to 50,000. Figure 16 and Figure 17 show that UQuadTKIS have 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 val-

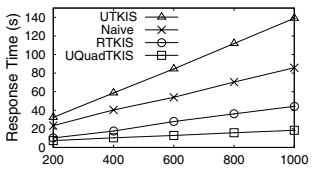


Figure 16: Diff. #facilities

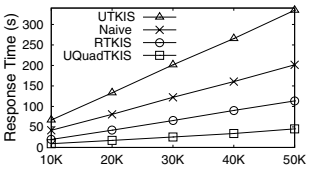
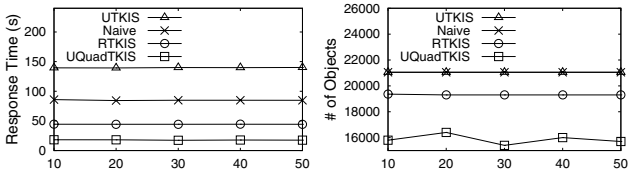


Figure 17: Diff. # objects

ues in Figure 18, which indicates that the performance of all algorithms are not very sensitive to the k value.



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

Figure 18: Diff. k

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 the *filtering* and *refinement* algorithms proposed in the paper 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.

6. CONCLUSION

In this paper, we investigate the problem of finding top k most influential facilities over a set of uncertain objects. Based on a new ranking model, we develop two 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 the paper to significantly improve the performance of the *filtering* and *refinement* algorithms. Our experiments convincingly demonstrate the effectiveness and efficiency of our techniques.

Acknowledgement. Ying Zhang is supported by ARC DP110104880 and UNSW ECR grant PS27476. Wenjie Zhang is supported by ARC DP120104168 and DE120102144. Xuemin Lin is supported by ARC DP0987557, ARC DP110102937, ARC DP120104168 and NSFC61021004.

7. REFERENCES

[1] T. Bernecker, T. Emrich, H.-P. Kriegel, M. Renz, S. Zankl, and A. Züfle. Efficient probabilistic reverse nearest neighbor query processing on uncertain data. *PVLDB*, 4(10):669–680, 2011.

[2] U. C. Bureau. `http://www.census.gov/geo/www/tiger`.

[3] M. A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei. Probabilistic reverse nearest neighbor queries on uncertain data. *IEEE Trans. Knowl. Data Eng.*, 22(4):550–564, 2010.

[4] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Querying imprecise data in moving object environments. *IEEE Trans. Knowl. Data Eng.*, 16(9):1112–1127, 2004.

[5] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, pages 305–316, 2009.

[6] R. A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 1974.

[7] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD Conference*, pages 47–57, 1984.

[8] M. Hua, J. Pei, W. Zhang, and X. Lin. Ranking queries on uncertain data: a probabilistic threshold approach. In *SIGMOD Conference*, pages 673–686, 2008.

[9] J. Huang, Z. Wen, J. Qi, R. Zhang, J. Chen, and Z. He. Top-k most influential locations selection. In *CIKM*, pages 2377–2380, 2011.

[10] F. Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In *SIGMOD Conference*, pages 201–212, 2000.

[11] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *VLDB J.*, 20(2):249–275, 2011.

[12] X. Lian and L. Chen. Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data. *VLDB J.*, 18(3):787–808, 2009.

[13] A. Papadopoulos and Y. Manolopoulos. Performance of nearest neighbor queries in r-trees. In *ICDT*, 1997.

[14] M. Shaked and J. G. Shanthikumar. *Stochastic Orders and Their Applications*. Academic Press, 1994.

[15] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. E. Hambrusch. Indexing uncertain categorical data. In *ICDE*, pages 616–625, 2007.

[16] M. A. Soliman, I. F. Ilyas, and K. C.-C. Chang. Top-k query processing in uncertain databases. In *ICDE*, pages 896–905, 2007.

[17] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *VLDB*, pages 922–933, 2005.

[18] Y. Theodoridis. The r-tree-portal. `http://www.rtreeportal.org`, 2003.

[19] R. C.-W. Wong, M. T. Özsu, A. W.-C. Fu, P. S. Yu, L. Liu, and Y. Liu. Maximizing bichromatic reverse nearest neighbor for lp -norm in two- and three-dimensional spaces. *VLDB J.*, 20(6), 2011.

[20] T. Xia, D. Zhang, E. Kanoulas, and Y. Du. On computing top-t most influential spatial sites. In *VLDB*, pages 946–957, 2005.

[21] D. Yan, R. C.-W. Wong, and W. Ng. Efficient methods for finding influential locations with adaptive grids. In *CIKM*, pages 1475–1484, 2011.

[22] X. Zhang and J. Chomicki. On the semantics and evaluation of top-k queries in probabilistic databases. In *ICDE Workshops*, pages 556–563, 2008.

[23] Y. Zhang, X. Lin, G. Zhu, W. Zhang, and Q. Lin. Efficient rank based knn query processing over uncertain data. In *ICDE*, pages 28–39, 2010.

[24] Y. Zhang, W. Zhang, Q. Lin, and X. Lin. Effectively indexing the multi-dimensional uncertain objects for range searching. In *EDBT*, 2012.

[25] K. Zheng, Z. Huang, A. Zhou, and X. Zhou. Discovering the most influential sites over uncertain data: A rank based approach. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99):1, 2011.