Discovering the Most Influential Sites over Uncertain Data: A Rank-Based Approach

Kai Zheng, Zi Huang, *Member*, *IEEE*, Aoying Zhou, *Member*, *IEEE*, and Xiaofang Zhou, *Member*, *IEEE*

Abstract—With the rapidly increasing availability of uncertain data in many important applications such as location-based services, sensor monitoring, and biological information management systems, uncertainty-aware query processing has received a significant amount of research effort from the database community in recent years. In this paper, we investigate a new type of query in the context of uncertain databases, namely *uncertain top-k influential sites query* (UTkIS query for short), which can be applied in a wide range of application areas such as marketing analysis and mobile services. Since it is not so straightforward to precisely define the semantics of top$k$ query with uncertain data, in this paper we introduce a novel and more intuitive formulation of the query on the basis of *expected rank* semantics. To address the efficiency issue caused by possible worlds exploration, we propose effective pruning rules and a divide-and-conquer paradigm such that the number of candidates as well as the number of possible worlds to be considered can be significantly reduced. Finally, we conduct extensive experiments on real data sets to verify the effectiveness and efficiency of the new methods proposed in this paper.

Index Terms—Uncertain data, reverse nearest neighbor query, top-k query

1 Introduction

Query processing over uncertain data has gained a lot of attentions from the database community recently, due to the imprecise nature in the data generated from a variety of real-world applications, such as sensor network, location-based service, market analysis, and the like. Generally, two types of uncertain data have been considered in previous work according to the cause of uncertainty. The first type is usually caused by the limitations of data collection techniques, such as data transmission delay, measurement accuracy, etc. For example, in sensor networks, collected sensor data might be distorted by environmental factors or pocket losses, and thus deviating from their actual values. For the second type, data uncertainty is caused by the multiple values that inherently exist in an object. The *NBA data set* [1] is a typical example of such a database, in which the performance of each player varies in different games. Due to its essential difference with precise data, traditional query processing techniques cannot be applied to handle uncertain data. Therefore, it is crucial to design novel approaches to efficiently answer queries over uncertain data and produce meaningful results. Up to now, many types of queries have been studied in the context of uncertain databases, such as range query [2], [3], [4], [5], (K-)NN query [3], [6], [7], [8], RNN query [9], (reverse) skyline query [1], [10], and similarity join [11], [12].

In this paper, we study another interesting query, finding top-k influential sites (TkIS for short), in the context of uncertain databases. TkIS [13] extends the *bichromatic reverse nearest neighbor (RNN) query* [14] in the sense that, instead of retrieving the RNNs for a particular query site, it returns the top-k sites which own the most RNNs (or most influential). More formally, given a spatial query region Q and a set of weighted objects, TkIS finds the top-k sites inside Q having the largest influences, where the *influence of a site s* is defined to be the sum of weights of the RNNs of s. Consider Fig. 1a as an example where the solid and blank circles represent sites and objects with equal weight, and query region is the entire space. By definition, the top-1 influential site should be s_1 whose influence is three. TkIS query is very useful in applications like mobile service (finding the wireless stations that have the most mobile users around) and market analysis (finding the super-markets that have the most residential buildings nearby).

By careful investigations, we observe that TkIS query is also meaningful in the scenarios where the objects are uncertain. More specifically, in our new problem settings, we assume that each object is characterized by multiple instances as in [1], and the sites remain deterministic. Fig. 1b exemplifies this setting, in which there are four sites and three uncertain objects. In the sequel, we showcase the usefulness of TkIS query in uncertain object set and identify the new challenges for answering this query.

1.1 Motivating Examples

As an extension of reverse nearest neighbor queries, TkIS query can also be useful in many real applications [14], [15], [16], such as decision support, resource allocation, army strategic planning, and mixed-reality games.

Decision support. Consider an example that a business owner plan to develop a new series of product (e.g., laptop). Before doing that, they want to know which of the new product is most likely to be popular among others. To this end, they represent their new products as points in a

- *K. Zheng, Z. Huang and X. Zhou are with the School of Information Technology and Electrical Engineering, University of Queensland, Brisbane 4072, Australia. E-mail: {kevinz, huang, zxf}@itee.uq.edu.au.*
- *A. Zhou is with the Software Engineering Institute, East China Normal University, Shanghai 200060, China. E-mail: ayzhou@sei.ecnu.edu.cn.*

Manuscript received 7 Nov. 2010; revised 18 Feb. 2011; accepted 3 May 2011; published online 26 May 2011.
Recommended for acceptance by Q. Li.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2010-11-0583.
Digital Object Identifier no.10.1109/TKDE.2011.121.

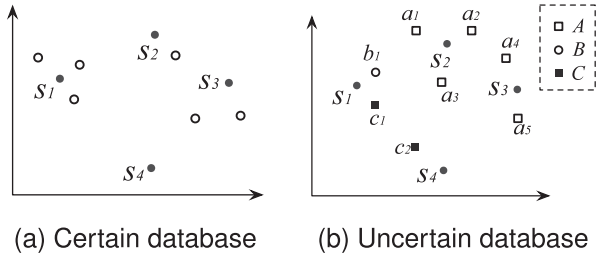


Fig. 1. TkIS query on certain and uncertain databases.

multidimensinal space, where each dimension corresponds to an attribute of this product (e.g., price range, color, weight, performance, etc.). Besides, they also need to know what people like, i.e., their preferences. However, the preferences are very subjective and usually equivocal in reality. A practical way to capture the preferences is design some questionnaire form and collect the feedback. Another approach is to infer the preference from previous purchase records. But, by either means, the derived preferences may not be precisely unique, since multiple purchase records may be kept for the same person. Motivated by this, we can naturally model the preferences as uncertain objects with multiple instances, and a TkIS query can be issued to find the most popular products.

Resource allocation. Another application is to monitor the ships in an ocean for rescue tasks. A ship having rescue task may be interested in those ships that have itself as their nearest neighbors. The reason is that, in case those ships encounter emergency, the rescue ship is the closest one to reach them and save lives of crews. However, the positions of ships can be also uncertain, due to various reasons such as the changed moving speed, transmission delay, or even the accuracy of the positioning devices. In such a case, the location of each ship can be inferred from its previously reported position, moving direction, and maximum speed, which results in a constrained uncertain region. Practically, the probability density function (PDF) in this uncertain region is often unavailable explicitly. Instead, a set of samples are drawn or collected in the hope of approximating the PDF. Correspondingly, we can model the position of each ship as a set of multiple points as its instances. Then, a TkIS query can be issued to find the rescuing ships with the potentially heaviest workload, so that reallocation of resources can be arranged if necessary.

By treating each instance as a possible representative of the object, we can obtain the influence of each site in some *possible world*. Since the influence may be different across all the possible worlds, it can be represented as a random variable or uncertain value. Then, the TkIS query can be transformed to an uncertain top-k (ranking) problem in which the sites to be ranked have influences as their uncertain attributes.

Top-k query over uncertain data has been studied in few works [17], [18], [19], [20], [21], [22]. In [17], five important properties including *exact-k*, *containment*, *unique ranking*, *value invariance*, and *stability* are proposed to evaluate the "goodness" of the top-k semantics. This motivates us to apply the *expected rank* semantic proposed in [17] to rank the uncertain influences of sites in our problem, as it is the only existing top-k semantic which meets all those properties. However, as we shall see in Section 3, directly applying the techniques, AE-rank [17], does not lead to a correct solution because the uncertain influences of sites are not *independent of each other*. Therefore, in this paper, we aim to develop correct and IO/CPU efficient algorithm to find the top-k influential sites based on their expected ranks.

Expected rank versus expected score. Another simple approach for ranking the sites with uncertain influences is to just compute the expected score (influence) of each site, and rank by this score, then take the top-k. It is easy to verify that such an approach directly implies exact-k, containment, unique ranking, and stability. However, as stated in [17], it violates value invariance property. Consider a site which has very low probability but a score that is orders of magnitude higher than others—then it gets to the top of the ranking since it has the highest expected score, even though it is unlikely. But, if we reduce this score to being just greater than the next highest score, this site will drop down the ranking. On the contrary, expected rank can avoid this problem since it relies on the ranks of the sites rather than the values in each possible world. Therefore, it can generally produce more stable results than the expected scores.

1.2 Contributions

To summarize, we make the following major contributions in this paper:

- We identify the potential application and usefulness of a novel query, *uncertain top-k influential sites* (UTkIS) query, in the context of uncertain databases.
- By reviewing the properties of different uncertain top-k semantics, we adopt the *expected rank* [17] as the ranking function to define the order of the sites with probabilistic influences.
- To remedy the efficiency issues caused by the correlation among the sites' influences, we propose a general filter-refine style approach which includes efficient PRNN search, effective pruning schemes, and divide-and-conquer-based refinement. Moreover, though our algorithms are proposed for the UTkIS query, they can be applied to other problems which adopt the expected rank as their ranking function.
- We present experimental observations which demonstrate the benefits of our proposed methodologies and optimizations.

The rest of this paper is organized as follows: Section 2 briefly reviews the related research efforts. In Section 3, we gather necessary background and formally defines the UTkIS query. Section 4 overviews our three-phase approach for processing the query, the details of which are elaborated in Sections 5, 6, and 7, respectively. Section 8 presents our experimental observations. Finally, in Section 9, we give some concluding remarks and outline directions for future work.

2 Related Work

The potential application and challenges of uncertain databases have inspired numerous work on this topic, making it impossible to review them all with limited space.

Since our work aims at applying top-k semantics to solve a spatial query over uncertain data, we mainly review the previous work from two aspects: spatial queries and top-k queries.

2.1 Spatial Queries over Uncertain Data

Many important spatial queries as well as their processing algorithms have been extended to uncertain databases. Among those work, nearest neighbor query is still the most flourishing topic. Particularly, Cheng and Chen [2] is the first to propose the concept of *probabilistic nearest neighbor* (PNN) query, which returns the objects that have nonzero qualification probabilities to be the nearest neighbor of the query point. Another method for evaluating a PNN is proposed in [8], where each object is represented as a set of points sampled from the object's continuous pdf. In order to improve the evaluation of qualification probabilities, Cheng et al. [6] propose a variant of 1-PNN that uses a probability threshold as an answering criterion, and have developed efficient verification methods for deriving lower and upper bounds of an object's qualification probabilities. To generalize 1-PNN to k-PNN ($k > 1$), Cheng et al. [7] propose *probability threshold k-NN* (T-k-PNN) query, which returns a k-subset whose qualification probability is not less than T. More recently, Zhang et al. [23] employ a rank-based approach to process k-PNN query, where k closest objects are returned according to their *expected ranks*.

In addition to nearest neighbor queries, more complex queries over uncertain data have also been proposed such as reverse nearest neighbor queries [9], group nearest neighbor queries [24], skyline queries [1], reverse skyline queries [10] and so on. Besides, novel indexing structures like U-tree [5] are also developed to facilitate processing probabilistic queries.

Inspired by these work, we identify the application and usefulness of another interesting query, TkIS query, in the context of uncertain databases and propose efficient algorithms to answer it.

2.2 Top-k Queries over Uncertain Data

Top-k query is an important topic in traditional databases. However, under the setting of uncertain data, the answering criterion is not straightforward, triggering different top-k semantics and algorithms proposed. The work in [20] is the first to identify the importance of top-k query processing in uncertain database and to propose methods to address it. In particular, they present two different definitions for top-k queries, U-Top$k$ and U-kRanks. The first one returns the set of tuples with the highest aggregated probability to be the top-k tuples across all possible worlds, while the tuple returned by the second query is the most probable tuple to appear at a given rank over all possible worlds. Their algorithm maps each configuration to a node of a super large graph and search over this graph with some generic, A*-like approach. Although this model can capture any correlation between tuples, the cost is exponential in both space and time. Later, Yi et al. [21] follow up this definition and propose more efficient algorithms under the x-relation model. Other than that, Zhang and Chomicki [22] propose a Global-Top$k$ semantics, which returns k highest ranked tuples according to their probability of being in the top-k answers over all possible worlds. Hua et al. [25] present a different approach called *probabilistic threshold top-k query*, which finds the set of records where each takes a probability of at least T to be in the top-k lists in the possible worlds. Recently, Cormode et al. [17] propose *expected rank*, which provably satisfies a set of nice properties of ranking queries over certain data. Ge et al. [18] define another new top-k semantics, called *c-Typical-Topk*, which returns c top-k vectors such that the actual top-k result (drawn according its distribution) is close to at least one of the c vectors.

Despite of so many proposals of top-k semantics over uncertain data, few work has applied them to other problems. In this sense, our work shares the motivation in common with [23], which also adopts the expected rank to model kNN query. Whereas in our paper, we focus on a entirely different query that is computationally more intensive than the nearest neighbor queries and thus requires the processing algorithms to be designed more dedicatedly.

TABLE 1
Summary of Notations

Notation	Definition
U	uncertain object
u	instance of uncertain object
s	site
$Pr_s(U)$	the probability of U being RNN of s
$PRNN(s)$	the probabilistic RNN set of s
$I(s)$	probabilistic influence of s
W	possible world
$\mathcal{W}$	the set of all possible worlds
$r_W(s), rank_W(s)$	the rank of s in possible world W
$er(s)$	the expected rank of s
$ler_P(s)$	local expected rank of s in partition P

3 Problem Statement

In this section, we briefly introduce the definition of expected rank and the list of top-k properties it meets. Then, we describe the uncertainty model and define the uncertain TkIS query based on expected rank. The notations and their definitions used in the paper is summarized in Table 1.

3.1 Preliminary: Expected Rank

We denote the uncertain relation as $\mathcal{D}$. In the attribute-level uncertainty model, an uncertain relation is instantiated into a *possible world* by taking one independent value for each tuple's uncertain attribute according to its distribution. Denote a possible world as W and the value for t_i's uncertain attribute in W as $t_i(W)$, then the probability that W occurs is $Pr(W) = \prod_{j=1}^{N} Pr[t_j = t_j(W)]$.

Definition 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.,*

$$rank_W(t_i) = |\{t_j \in W \mid t_j(W) > t_i(W)\}|.$$

The expected rank *of a tuple t_i is defined to be*

$$er(t_i) = \sum_{W \in \mathcal{W}} Pr(W) \cdot rank_W(t_i).$$

Among all the uncertain top-k semantics proposed recently, expected rank is the only one that meets all the following properties migrated from certain databases:

- *Exact-k:* The top-k list should contain exactly k items;
- *Containment:* The top-$(k+1)$ list should contain all items in the top-k;
- *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.
- *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;
- *Stability:* Making an item in the top-k list more likely or more important should not remove it from the list.

Remark. With uncertain data, there are two distinct orders to work with: score and probability. There are many possible ways of combining these two, leading to quite different results. Also, as pointed out in [26], ranking in probabilistic databases is inherently a multicriteria optimization problem. So, there seems no "one-fit-all" top-k definition for uncertain data and semantic comparison of them is out of the scope of our work. In this paper, we focus on efficient solution for finding the top-k influential sites given an uncertain object set based on the expected ranks of sites.

3.2 Uncertainty Model

In previous work of uncertain databases, two major classes of uncertainty models are assumed: tuple- and attribute-uncertainty. We use the attribute-uncertain model throughout the paper. In our problem, there are two types of data, namely *site* and *object*. As the sites usually represent servers or buildings whose locations do not frequently change, we keep the site data set deterministic (precise points). On the other hand, we model an uncertain object U as a set of points (or instances) in the data space, denoted by $U = \{u_1, u_2, \ldots, u_l\}$. The reason we adopt the discrete form to represent uncertain objects rather than *probability density function* is that, as discussed in [1], [23], the PDF is often explicitly unavailable in practice and thus approximated by drawing instance samples.

According to our uncertainty model, an uncertain object may be *influenced* by multiple sites instead of one as the deterministic case, where by "A influences B" we mean that B is a *reverse nearest neighbor* of A. In other words, for a site s, each of its RNNs is *probabilistic*. We let $PRNN(s)$ to be this probabilistic RNN set of s, i.e., $PRNN(s) = \{(U, Pr_s(U)) \mid Pr_s(U) > 0\}$, where $Pr_s(U)$ is the probability of U being influenced by s and can be computed as

$$Pr_s(U) = \sum_{u \in U, u \in RNN(s)} Pr(u).$$

Consequently, the influence of a site s, denoted as $I(s)$, is not a unique value any more. Instead, $I(s)$ becomes a random variable whose value can vary depending on the distribution of the $PRNN(s)$. The *probability mass function (pmf)* of $I(s)$ can be computed by the following equation:

$$\begin{aligned} f_s(x) &= Pr[I(s) = x] \\ &= \sum_{\beta \in \alpha_s(x)} \prod_{U \in \beta} Pr_s(U) \times \prod_{U \in PRNN(s) \setminus \beta} (1 - Pr_s(U)), \end{aligned} \quad (1)$$

where $\alpha_s(x)$ is all the subsets of $PRNN(s)$ in which the sum of weights of the objects are equal to x, i.e.,

$$\alpha_s(x) = \left\{ \forall A \subseteq PRNN(s) \mid \sum_{U \in A} w(U) = x \right\}.$$

Example 1. Consider the example illustrated in Fig. 1b where the weights of all objects are assumed to be 1. Then, the PRNN and probabilistic influence of each site are listed in the following table.

Site	PRNN	pmf
s_1	$(B, 1.0), (C, 0.5)$	$f(1) = 0.5, f(2) = 0.5$
s_2	$(A, 0.6)$	$f(0) = 0.4, f(1) = 0.6$
s_3	$(A, 0.4)$	$f(0) = 0.6, f(1) = 0.4$
s_4	$(C, 0.5)$	$f(0) = 0.5, f(1) = 0.5$

3.3 Uncertain $\mathrm{T}k\mathrm{IS}$ Query

By viewing the influence of a site as its uncertain attribute, we can naturally adopt the expected rank to define the uncertain $\mathrm{T}k\mathrm{IS}$ query.

Given a set of sites S, a set of uncertain objects $\mathcal{U}$ and a query region Q, we define all sites inside Q as the candidate sites, i.e., $C_s = \{s \mid s \in Q\}$, and all objects influenced by candidate sites as the candidate objects $C_u = \{U \mid \exists s \in C_s, U \in PRNN(s)\}$. 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, we observe that for an uncertain object, all the instances which are the reverse nearest neighbors of the same site are equivalent with respect to computing the influences. Therefore, we can regard those instances as a group so that the total number of possible worlds will be reduced. For a particular possible world W, the *rank* of a site $s \in C_s$ in W is defined to be the number of candidate sites whose influence is greater than s, i.e.,

$$r_W(s) = |\{s' \in C_s \mid I_W(s') > I_W(s)\}|,$$

where $I_W(s)$ is the influence of site s in possible world W. The *expected rank* of a site is the expectation of its ranks across all possible worlds, i.e.,

$$er(s) = \sum_{W \in \mathcal{W}} r_W(s) \times Pr(W). \quad (2)$$

The smaller $er(s)$ is, the higher s ranks.

Definition 2. *Given a set of sites S, a set of uncertain objects $\mathcal{U}$, a query region Q, and a natural number k, the* uncertain top-k influential site query *returns the top-k sites in Q according to their expected ranks.*

Example 2. Continuing the Example 1, we aim to get the expected ranks of all the sites. The possible worlds, as well as their probabilities and rank of each site are listed in the table below.

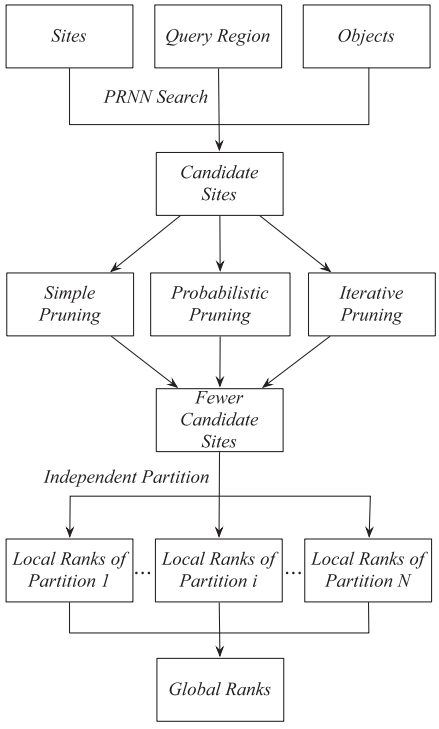


Fig. 2. Approach overview.

	Possible world	Probability	Ranks
W_1	$A = (a_1, a_2, a_3)$, $B = b_1$, $C = c_1$	$0.5 \times 0.6 = 0.3$	$s_1 > s_2 > s_3 = s_4$
W_2	$A = (a_4, a_5)$, $B = b_1$, $C = c_1$	$0.5 \times 0.4 = 0.2$	$s_1 > s_3 > s_2 = s_4$
W_3	$A = (a_1, a_2, a_3)$, $B = b_1$, $C = c_2$	$0.5 \times 0.6 = 0.3$	$s_2 = s_4 = s_1 > s_3$
W_4	$A = (a_4, a_5)$, $B = b_1$, $C = c_2$	$0.5 \times 0.4 = 0.2$	$s_3 = s_4 = s_1 > s_2$

Then, the expected ranks of each site can be computed by (2):

$$er(s_1) = 0.3 \times 0 + 0.2 \times 0 + 0.3 \times 0 + 0.2 \times 0 = 0$$
$$er(s_2) = 0.3 \times 1 + 0.2 \times 2 + 0.3 \times 0 + 0.2 \times 3 = 1.3$$
$$er(s_3) = 0.3 \times 2 + 0.2 \times 1 + 0.3 \times 3 + 0.2 \times 0 = 1.7$$
$$er(s_4) = 0.3 \times 3 + 0.2 \times 3 + 0.3 \times 0 + 0.2 \times 0 = 1.5.$$

To process the UTkIS query, it seems natural to adopt the *A-ERank* algorithm [17] since it can evaluate the expected ranks in $O(N \log N)$ time without exploring the possible worlds. However, their algorithms are based on the assumption that the attributes of tuples to be sorted are independent of each other, which does not hold in our case. We can use a simple example to illustrate this property. Consider the probabilistic influences calculated in the Example 1, and we have $Pr[I(s_1) = 4] \times Pr[I(s_4) = 3] = 25\%$. But actually, it is impossible for $I(s_1)$ to be 4 and $I(s_4)$ to be 3 at the same time, i.e., $Pr[I(s_1) = 4 \cap I(s_4) = 3] = 0$, since they share a common object C as PRNN. Therefore, to guarantee the correctness of the algorithm, we have to evaluate the ranks of the sites in every possible world, the cost of which is prohibitive even when the data set is fairly large. This requires us to design more elaborate algorithms, so that the query can be answered more efficiently.

4 Approach Overview

To answer the UTkIS query, we propose a pruning-refinement style approach that consists of three steps. In this section, we just give an overview of the approach, leaving more details elaborated in the following sections.

4.1 PRNN Search

Naturally, to find the most influential sites, we need to know the influences of the sites, which requires us to retrieve the $PRNN$ set for each site inside Q. As a common strategy for processing RNN queries [14], [27], [28], [16], [29], we can index the instances of objects with R-tree and then issue an RNN query for each site against the whole instance set. However, since the size of instance set is usually large, the cost of this method can be very high. In Section 5, we will present several optimization mechanisms to improve its performance.

4.2 Pruning

Since only the top-k sites are needed, where k is usually small compared to the total number of sites, it is not necessary to exhaustively evaluate the expected ranks for all sites. Motivated by this, we develop several pruning rules to eliminate the sites with no hope to be top-k. Specifically, the *simple pruning (SP) rule* only makes use of the extreme values of the sites' influence hence has limited pruning power. The *probabilistic pruning (PP) rule* is based on the more sophisticated probabilistic bounds of the influence. We also exploit Hoeffding's inequality to approximate the probabilistic bounds efficiently. Finally, an *iterative pruning* (IP) algorithm is proposed which will iteratively apply the probabilistic pruning rule to enhance the pruning effects. According to the empirical study of [17], though it is impossible to obtain a precise order on their final ranks without inspecting all sites in the original candidate set, the expected rank of each site in the curtailed set is an excellent surrogate. Hence, we return the top-k of these as the query result.

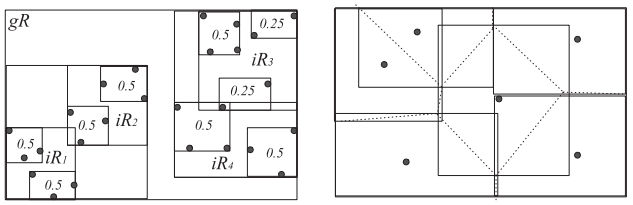
4.3 Rank Evaluation

After all, the expected ranks of the sites which cannot be pruned still need to evaluated. Since straightforwardly unfolding possible worlds on the whole candidate set is very time consuming, we propose to divide all the candidate sites into *independent partitions*, in which *local expected ranks* can be calculated more efficiently. Then, the *global expected rank* can be obtained by summing up its local ranks in each partition. In addition, by using the *partial expected ranks*, we can eliminate more sites and diminish the possible worlds further.

5 PRNN Search

Although the general approach follows the *branch-and-bound* paradigm with the facilitation of R-tree, we propose several optimizations to improve the efficiency of this step.

First, we assume the uncertain regions of objects are indexed by a *global R-tree* (gR). To avoid loading all instances of an object into memory during computation, its instances are also organized by an *instance aR-tree* (iR), with the fraction of instances recorded in every node. Each leaf node of gR contains a pointer to an iR of the object.



(a) Global and instance R-tree (b) Site R-tree

Fig. 3. Indexing of objects and sites.

Voronoi Diagram is usually used to speed up the determination of the nearest point to a given query. Ideally, if we precompute and store the *Voronoi Cell* of all the sites, determining whether a point belongs a site's RNN just requires a *Point-in-Polygon Test*. However, explicitly storing the original Voronoi Cells needs $O(N^{\lceil d/2 \rceil})$ space that is exponential with the object dimension. Therefore, we only keep the MBR of each Voronoi Cell and index them with another *site R-tree* (sR). Fig. 3 illustrates the object and site R-trees.

For a given region Q, we first find all the *candidate sites* C_s by issuing a window query in the sR to get all the sites inside Q. For each site $s \in C_s$, we search the gR for the *candidate objects* of s whose uncertain region (MBR of leaf node) intersects with the MBR of s. From the property of Voronoi Cell, only those object are possible to be the PRNN of s. The last step is to refine the qualification probability $Pr_s(U)$ for all candidate objects. In order to avoid repetitive search over the instance R-tree of a particular object, we maintain a set $IS(U)$ of *influencing site* for each candidate object U, consisting of sites having U as their candidate in previous step. Then, we traverse the iR of U from root to leaf once. During this process, if the encountered node falls inside the MBR of some site $s \in IS(U)$ only, all the instances inside this node are guaranteed to the RNNs of s; otherwise, we need to recursively move downwards to the child nodes. When the leaf node is reached and still cannot be resolved, we have to probe the instances of this node and calculate their distances with the sites of $IS(U)$. Since the number of objects is much less than the number of instances, this approach can achieve better performance by avoiding repetitive search over the entire instance set.

TABLE 2
Example of Pruning Scheme

s	$PRNN(s)$
A	80%, 80%, 80%, 100%, 100%
B	50%
C	40%, 50%, 60%
D	10%, 15%, 20%
E	80%, 80%, 80%, 80%, 100%, 100%

6 Pruning Scheme

It is prohibitively expensive to unfold all the possible worlds and calculate the expected rank for each candidate site. In this section, we investigate several novel pruning schemes that can exclude a portion of sites without knowing their expected ranks.

6.1 Simple Pruning

By scanning the PRNN set of each candidate site, we can derive the *simple upper and lower bounds* of the influence by the following equation:

$$I^+(s) = \sum_{U \in PRNN(s)} w(U), \qquad I^-(s) = \sum_{\{U \in PRNN(s) | Pr_s(U)=1\}} w(U). \tag{3}$$

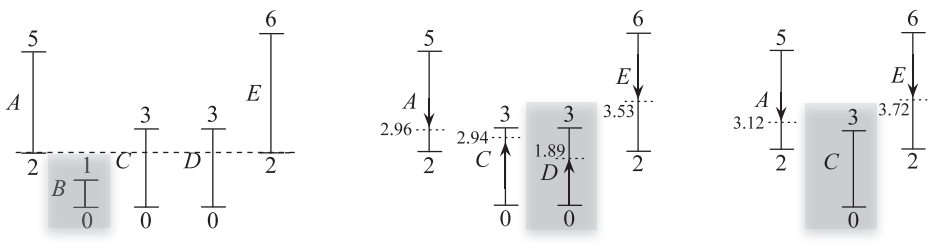
For any two sites s and s', if $I^-(s') > I^+(s)$, then in every possible world, the influence of s will be less than s'; hence, the expected rank of s will be larger than s'. Therefore, with the simple bounds of influences, we can immediately eliminate some sites that are impossible to be the results by the following lemma.

Lemma 3. *A site s cannot be a result of* UTkIS *if there exist at least k sites s' satisfying $I^-(s') > I^+(s)$.*

Example 3. Suppose we want to find the top-2 among the five candidate sites listed the Table 2. Note that we assume the weights of objects are 1, so only the probability of the PRNN is given for the sake of briefness. The simple upper and lower bounds of the influences are shown in Fig. 4a. By Lemma 3, site B can be pruned immediately since its upper bound is lower than the lower bound of site A and E. No other sites can be pruned any more by simple pruning method.

6.2 Probabilistic Pruning

In general, the simple pruning rule is only able to eliminate a small number of disqualifying sites, since the strict



(a) Simple Pruning (b) Probabilistic Pruning (c) Iterative Pruning

Fig. 4. Pruning methods.

comparison criteria limits its pruning power at large. Is it possible to relax the Lemma 3 "a little bit"? In other words, even if $I^+(s) \geq I^-(s')$, can we still prune s without knowing their exact ranks? We first introduce the following theorem which states that s may still rank lower than s' if the chance for $I(s)$ exceeding $I(s')$ is small enough.

Theorem 4. *Given a set of candidate sites C_s, $er(s)$ is guaranteed to be larger than $er(s')$ if the probability of $I(s)$ being greater than $I(s')$ is not above $\frac{1}{|C_s|}$, i.e.,*

$$Pr[I(s) > I(s')] \leq \frac{1}{|C_s|}.$$

Proof. First, we divide the set of all possible worlds $\mathcal{W}$ into two groups, $\mathcal{W}_{s'}$ in which $I(s') > I(s)$ and $\mathcal{W}_s$ in which $I(s') \leq I(s)$

- For each possible world $W \in \mathcal{W}_{s'}$, let $r_W(s')$ denote the rank of s' in W, then $r_W(s) \geq r_W(s') + 1$.
- For each possible world $W \in \mathcal{W}_s$, $r_W(s') \leq |C_s| - 1$ and $r_W(s) \geq 0$.

Then, we can get the upper bound of the $er(s')$ and the lower bound of the $er(s)$ as follows:

$$\begin{aligned} er(s') &\leq er(s')^+ \\ &= \sum_{W \in \mathcal{W}_{s'}} Pr(W) \cdot r_W(s') + \sum_{W \in \mathcal{W}_s} Pr(W) \cdot (|C_s| - 1) \\ er(s) &\geq er(s)^- \\ &= \sum_{W \in \mathcal{W}_{s'}} Pr(W) \cdot (r_W(s') + 1) + \sum_{W \in \mathcal{W}_s} Pr(W) \cdot 0 \\ &= \sum_{W \in \mathcal{W}_{s'}} Pr(W) \cdot (r_W(s') + 1). \end{aligned}$$

Since

$$\begin{aligned} Pr[I(s) \geq I(s')] &\leq \frac{1}{|C_s|} \\ \sum_{W \in \mathcal{W}_s} Pr(W) &\leq \frac{1}{|C_s|} \\ \sum_{W \in \mathcal{W}_{s'}} Pr(W) &> 1 - \frac{1}{|C_s|} \end{aligned}$$

$$\begin{aligned} er(s')^+ &= \sum_{W \in \mathcal{W}_{s'}} Pr(W) \cdot r_W(s') + (|C_s| - 1) \sum_{W \in \mathcal{W}_s} Pr(W) \\ &\leq \sum_{W \in \mathcal{W}_{s'}} Pr(W) \cdot r_W(s') + (|C_s| - 1) \cdot \frac{1}{|C_s|} \\ &= \sum_{W \in \mathcal{W}_{s'}} Pr(W) \cdot r_W(s') + 1 - \frac{1}{|C_s|} \\ &< \sum_{W \in \mathcal{W}_{s'}} Pr(W) \cdot r_W(s') + \sum_{W \in \mathcal{W}_{s'}} Pr(W) \\ &= \sum_{W \in \mathcal{W}_{s'}} Pr(W) \cdot (r_W(s') + 1) \\ &= er(s)^- \end{aligned}$$

So, $er(s') < er(s)$, which means the expected rank of s' is sure to be smaller than that of s. □

According to Theorem 4, we can define the *probabilistic upper and lower bounds* for the influence of a site.

Definition 5. *For any candidate site $s \in C_s$, its probabilistic upper and lower bound can be defined as*

$$\begin{aligned} I^{\uparrow}(s) &= \min\left\{ v \in \mathbb{R} | Pr(I(s) \geq v) \leq \frac{1}{|C_s|} \right\} \\ I^{\downarrow}(s) &= \max\left\{ v \in \mathbb{R} | Pr(I(s) \leq v) \leq \frac{1}{|C_s|} \right\}. \end{aligned} \tag{4}$$

Corollary 6. *Given two sites s and s', it is guaranteed that $er(s) > er(s')$, if they satisfy either 1) $I^-(s') > I^{\uparrow}(s)$, or 2) $I^{\downarrow}(s') > I^+(s)$.*

Proof. We first prove the case when they satisfy 1:

$$\begin{aligned} &Pr[I(s) > I(s')] \\ &= \int_{v \in \mathbb{R}} Pr[I(s) = v] Pr[I(s') < v \mid I(s) = v] dv \\ &= \int_{v \leq I^{\uparrow}(s)} Pr[I(s) = v] Pr[I(s') < v \mid I(s) = v] dv \\ &\quad + \int_{v > I^{\uparrow}(s)} Pr[I(s) = v] Pr[I(s') < v \mid I(s) = v] dv \\ &= 0 + \int_{v > I^{\uparrow}(s)} Pr[I(s) = v] Pr[I(s') < v \mid I(s) = v] dv \\ &\leq \int_{v > I^{\uparrow}(s)} Pr[I(s) = v] dv \\ &= Pr[I(s) \geq I^{\uparrow}(s)] \leq \frac{1}{|C_s|}. \end{aligned} \tag{5}$$

By analogy, the case 2 can be proved. □

Compared to the simple bounds, the probabilistic bounds are tighter since they do not bound all the possible influence values as the simple bounds do. Instead, they just bound the "most likely values."

Now, the only problem left is, given the $PRNN(s)$, how to calculate the *probabilistic bounds* in Definition 5. A straightforward method is to calculate the probability mass function of $I(s)$ in (1). To do that, we need to explore all the combinations of the elements in $PRNN(s)$, the complexity of which is exponential with the size of $PRNN(s)$. Actually, deriving the *pmf* of $I(s)$ can be inducted to the *subset-sum* problem which is a classic NP-complete problem [30]. So, we need to develop a more efficient way to compute these bounds. Fortunately, as these bounds are just used for pruning, it is not necessary to get their exact values. An approximated bounds can still guarantee the correctness of Corollary 6 as long as it is looser than the original one.

In the sequel, we aim to conservatively approximate $I^{\uparrow}(s)$ and $I^{\downarrow}(s)$ by applying Hoeffding's inequality.

Approximating $I^{\uparrow}(s)$ and $I^{\downarrow}(s)$ by tail bound. In probability theory, several *tail bounds* have been developed such as Markov inequality, Chebyshev inequality, Hoeffding inequality, Chernoff bound, etc., to estimate the probability on tail distribution of random variables. Generally, the more information (e.g., mean, variance) they require, the tighter bounds they give. In this paper, we adopt the Hoeffding inequality because it gives exponentially

decreasing bounds while Markov inequality and Chebyshev inequality only yield polynomial bounds. Besides, all the information required can be obtained in our problem.

Hoeffding inequality [31] gives an upper bound on probability for the sum of independent random variables to deviate from its expected value. Let $X_1, \ldots, X_n$ be *independent* and *almost surely bounded* random variables, i.e., for $1 \leq i \leq n$ that

$$Pr[X_i - E(X_i)] \in [a_i, b_i]) = 1.$$

Then, for the sum of these variables, $s = X_1 + \cdots + X_n$, we have the following inequalities for any positive values t,

$$\begin{aligned} Pr(s \geq E(s) + nt) &\leq exp\left(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right), \\ Pr(s \leq E(s) - nt) &\leq exp\left(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right). \end{aligned} \tag{6}$$

In the light of this theory, if we treat each object U in the PRNN set of site s as a random variable, which has two possible values, $w(U)$ and 0, with probabilities of $Pr_s(U)$ and $1 - Pr_s(U)$ respectively, $I(s)$ is actually the sum of these *independent and bounded random variables*, i.e., $I(s) = \sum_{U \in PRNN(s)} U$. We use $E(U)$ and $\delta(U)$ to denote the expected value and the difference of extreme values of U, then

$$\begin{aligned} E(U) &= Pr_s(U) \cdot w(U) \\ \delta(U) &= \begin{cases} w(U), & if\ 0 < Pr_s(U) < 1 \\ 0, & if\ Pr_s(U) = 1. \end{cases} \end{aligned} \tag{7}$$

By letting the right sides of (6) to be $\frac{1}{|C_s|}$, we have

$$nt = \delta[I(s)] \cdot \sqrt{\frac{\ln |C_s|}{2}}.$$

Then, the probabilistic bounds of $I(s)$ can be approximated by the following equations:

$$\begin{aligned} I^{\uparrow *}(s) &= E[I(s)] + \delta[I(s)] \cdot \sqrt{\frac{\ln |C_s|}{2}}, \\ I^{\downarrow *}(s) &= E[I(s)] - \delta[I(s)] \cdot \sqrt{\frac{\ln |C_s|}{2}}. \end{aligned} \tag{8}$$

Although $I^{\uparrow *}(s)$ and $I^{\downarrow *}(s)$ are just approximations, Corollary 6 can still apply, since the inequality $Pr[I(s) \geq I^{\uparrow *}(s)] \leq \frac{1}{|C_s|}$ and $Pr[I(s) \leq I^{\downarrow *}(s)] \leq \frac{1}{|C_s|}$ still stand. If, in some cases, the approximated upper (lower) bound is greater (less) than the simple upper (lower) bound, we use the simple upper (lower) bound to do pruning. Evaluation of (8) only requires one scan of the PRNN set for each site. So, the complexity is linear to the size of PRNN set. Once the approximated probabilistic bounds have been derived, we can treat them as simple bounds, and prune the site s whenever there are k sites s' such that $I^{\downarrow *}(s') > I^{\uparrow *}(s)$.

Example 4. Now, we apply the probabilistic pruning method to the four sites left from Example 3. By computing the $E[I(s)]$ and $\delta[I(s)]$ according to Fig. 2, and substituting $|C_s|$ with 4, we can get the probabilistic bounds for each site, as shown in Fig. 4b. Since $I^{\uparrow *}(D) < I^{-}(A)$ and $I^{\uparrow *}(D) < I^{-}(E)$, site D will be pruned by Corollary 6.

6.3 Iterative Pruning

By examining (8) carefully, we observe that the probabilistic bounds get improved as the number of candidate sites decreases. This motivates us to devise an iterative pruning algorithm, which can be described by the following steps:

1. The approximated probabilistic bounds are evaluated based on the cardinality of candidate set C_s.
2. Prune the disqualifying sites.
3. If any site gets pruned in step 2, $|C_s|$ will reduce. So, we go to step 1 to reevaluate the bounds based on the newly updated candidate set. Since the probabilistic bounds get tighter, more sites might be pruned. However, if no site gets pruned in this step, the algorithm can terminate.

Example 5. Continuing the Example 3, this time we use the iterative pruning algorithm to verify the advantages of our method. The first round is exactly the same as the Example 4 after which D gets pruned. Since the candidate set C_s has changed, we reevaluate the probabilistic bounds of the three sites left by substituting $|C_s|$ with 3. The updated probabilistic bounds are displayed in Fig. 4c. Then, site C can be safely removed, as $I^{\downarrow *}(A) > I^{+}(C)$ and $I^{\downarrow *}(E) > I^{+}(C)$, leaving A and E as the final results.

Correctness. Recall that according to the empirical study of [17], it is impossible to obtain a precise order on their final ranks without inspecting all sites in the original candidate set. In other words, in each iteration of pruning, the expected ranks of the sites may slightly fluctuate due to the change of the cardinality. However, we claim that our iterative pruning scheme is correct, in the sense that it will not prune any site that is the top-k result in the original data set. Formally, we will prove the following corollary.

Corollary 7. *A site that is removed by the iterative pruning method cannot belong to the top-k results of the original data set.*

Proof. Suppose in the ith iteration, s_1 is one of the pruned sites, s_2 is one of the top-k sites that contribute to prune s_1, and $|C|$ is the cardinality of the site set. Clearly, $er(s_2) < er(s_1)$. We will show that in the $i-1$th iteration, $er(s_2) < er(s_1)$ still holds. Now, we randomly restore one of the pruned sites s_3 back to the data set. According to the pruning rule, $Pr[I(s_2) > I(s_1)] < 1/|C|$, $Pr[I(s_3) > I(s_1)] < 1/(|C|+1)$. Then, we need to deal with the following possible cases:

1. $I(s_1) > I(s_2), I(s_3) > I(s_2)$: In this case, the expected ranks of s_1 and s_2 change to $er(s_1) + 0.5$ and $er(s_2) + 1$;
2. $I(s_1) > I(s_2), I(s_3) \leq I(s_2)$: In this case, the expected ranks of s_1 and s_2 remains the same;
3. $I(s_1) \leq I(s_2), I(s_3) > I(s_2)$: In this case, the expected ranks of s_1 and s_2 change to $er(s_1) + 1$ and $er(s_2) + 1$;

4. $I(s_1) \leq I(s_2), I(s_3) \leq I(s_2)$: In this case, the expected ranks of s_1 change to $er(s_1) + 0.5$.

To simplify the proof, we consider a situation which is the best for the rank of s_1 to be promoted in the $i-1$th iteration, that is $Pr[I(s_2) > I(s_1)] = 1/|C|$, $Pr[I(s_3) > I(s_1)] = 1/(|C|+1)$. By taking the four cases into consideration, the new expected ranks of s_1 and s_2 change to

$$er'(s_1) = er(s_1) + \frac{|C|^2 + |C| - 1}{2|C|(|C|+1)}$$
$$er'(s_2) = er(s_2) + \frac{1}{|C|+1}.$$

Now, we have

$$er'(s_1) - er'(s_2) = er(s_1) - er(s_2) + \frac{|C|^2 - |C| - 1}{2|C|(|C|+1)}.$$

So, $er'(s_1) > er'(s_2)$ holds for any $|C| > 1$. We can apply the above procedure while restoring more removed sites in previous iterations until the data set is recovered to the original one, which means s_1 cannot belong to the top-k in the original data set. □

7 Rank Evaluation

This phase evaluates the expected ranks for the sites that cannot be eliminated by the pruning phase. However, straightforwardly computing the expected ranks on the whole candidate set can be very time consuming, due to the exponential number of possible worlds. Suppose there are n objects in C_{obj} and each object is influenced by m sites on average, then the total number of possible worlds to be unfolded is about m^n, which can be extremely large even for a moderate n. So, in this section, we provide a divide-and-conquer algorithm to reduce the number of possible worlds. Based on that, a sorted access optimization is also proposed, in order to return the top-k without checking all sites.

7.1 Independent Partition

First, we define the *independent partition* as follows:

Definition 8. *Two sites s and s' are called* independent *if they do not share common objects in their PRNN sets. Given a candidate set C_{site}, its* independent partition *is a set of nonoverlapping groups of sites, i.e., $\mathcal{P}_{site} = \{P_1, P_2, \ldots, P_m\}$ such that any two sites from different groups are independent of each other.*

If we treat all candidate sites as the nodes of a graph, and link two nodes by an edge if they are not independent, partitioning all the sites is equivalent to finding all components (maximally connected subgraph) of the graph. For example, Fig. 5 illustrates a graph consisting of nine sites which initially cannot be partitioned. However, after the candidate selection phase, three sites get pruned (blank circle). By simple depth-first traversal of this graph, we can obtain three independent partitions (components).

The benefit of independent partition is that, we can calculate the expected rank of a site s in some partition P, called *local expected rank*, by exploring fewer possible worlds since the numbers of both sites and objects are smaller. Then, by following lemma, it is also simple to calculate the $er(s)$, once its local ranks in all partitions have been obtained.

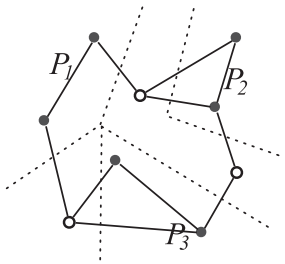


Fig. 5. Independent partition.

Lemma 9. *The expected rank of a site s is the sum of its local ranks in all partitions, i.e.,*

$$er(s) = \sum_{P \in \mathcal{P}_{site}} ler_P(s). \tag{9}$$

Proof. Given a possible world W, we decompose it into m elements, $W = \{W_1, \ldots, W_m\}$, where W_i corresponds to the possible world instantiated based on partition P_i. Since each partition is independent of others, we have $Pr(W) = Pr(W_1) \times \cdots \times Pr(W_m)$. Similarly, we can also represent the set of all possible worlds $\mathcal{W} = \{\mathcal{W}_1, \ldots, \mathcal{W}_m\}$. We let $r_{W_i}(s)$ to denote the local rank of s in partition P_i and possible world W_i.

The global expected rank of site s can be rewritten as

$$\begin{aligned} er(s) &= \sum_{W \in \mathcal{W}} Pr(W) \times r_W(s) \\ &= \sum_{W_1 \in \mathcal{W}_1, \ldots, W_m \in \mathcal{W}_m} (Pr(W_1) \times \cdots \times Pr(W_m)) \\ &\quad \times (r_{W_1}(s) + \cdots + r_{W_m}(s)) \\ &= \zeta_1 + \cdots + \zeta_m, \end{aligned} \tag{10}$$

where

$$\zeta_i = \sum_{W_1 \in \mathcal{W}_1, \ldots, W_m \in \mathcal{W}_m} (Pr(W_1) \times \cdots \times Pr(W_m)) \times r_{W_i}(s).$$

Without loss of generality, we investigate the term ζ_1, which can be rewritten as

$$\begin{aligned} \zeta_1 &= \sum_{W_1 \in \mathcal{W}_1} (Pr(W_1) \times r_{W_1}(s)) \\ &\quad \times \sum_{W_2 \in \mathcal{W}_1, \ldots, W_m \in \mathcal{W}_m} (Pr(W_2) \times \cdots \times Pr(W_m)) \\ &= ler_{P_1}(s) \times \sum_{W_2 \in \mathcal{W}_2} Pr(W_2) \times \cdots \times \sum_{W_m \in \mathcal{W}_m} Pr(W_m) \\ &= ler_{P_1}(s). \end{aligned} \tag{11}$$

Therefore,

$$er(s) = ler_{P_1}(s) + \cdots + ler_{P_m}(s).$$

□

Depending on whether the site s belongs to the partition P or not, two cases must be treated differently:

- If $s \in P$, the local expected rank of s have to be calculated by exploring the possible worlds based on the sites in partition P.
- If $s \notin P$, since s is independent of all the sites in P, we can apply the following equation to evaluate the local expected rank of s:

$$ler_P(s) = \sum_{s' \in P} Pr[I(s') > I(s)]. \quad (12)$$

Let U be the universe of all the influence values in P. We can precompute $q(t) = \sum_{v \in U, v > t} Pr(v)$ for all $t \in U$ by a linear pass after sorting the elements of U. Then, (12) can be evaluated within constant time, by $ler_P(s) = \sum_{t \in I(s)} Pr(t) \times q(t)$.

By dividing the candidate set into several partitions with smaller sizes, the number of possible worlds we need to explore can be considerably reduced. We can show the benefit gain by a simple analysis. If, after independent partitions, the n candidate objects are uniformly divided into d groups, then the total number of possible worlds will be $d \times m^{n/d}$. To make a concrete comparison, we assume $n = 30, m = 2, d = 3$. Initially, $|\mathcal{W}| = 2^{30} \approx 10^9$, but after the partitioning, $|\mathcal{W}| = 3 \times 2^{10} < 10^4$, which is less than 0.01 percent of the original size.

7.2 Partial Expected Ranks

For some site $s \in P_i$, its local expected ranks in any partition $P_j \in \mathcal{P}/P_i$ can be evaluated efficiently without unfolding the possible worlds. We call the sum of the local expected ranks of s in all partition except P_i the *partial expected rank* of s, denoted by $per(s)$

$$per(s) = \sum_{P_j \in \mathcal{P}/P_i} ler_{P_j}(s).$$

Following Lemma 9, we can immediately get that the partial expected rank of s is a lower bound of its actual expected rank, i.e.,

$$er^-(s) = per(s) \leq er(s).$$

To get an upper bound of $er(s)$, we estimate its local expected rank in P_i in the most pessimistic way, which is the largest possible rank $ler^+_{P_i}(s) = n_i - 1$, where n_i is the number of sites in P_i. So,

$$er^+(s) = per(s) + n_i - 1 \geq er(s).$$

These bounds lead immediately to the following algorithm:

1. Divide the candidate set C_s into independent partitions, and calculate $er^-(s)$ and $er^+(s)$ for each site $s \in C_s$.
2. We find the kth smallest $er^+(s)$ value, denoted by er^+_k and compare this to every $er^-(s)$. If $er^-(s) > er^+_k$, we know for sure that site s cannot be the top-k hence can be pruned.
3. If some sites get pruned in step 2, we go to step 1; otherwise, we have to evaluate the local expected rank for each site on its own partition.

TABLE 3
Parameter Settings

Definition	Default setting
Distribution of instances	Constrained Normal
Query region (% of whole space)	5%
Radius of uncertain region	0.3
Number of instances	100
Request number of sites	8

Improving er^+ and er^- by sampling possible worlds. The estimation for the upper and lower bounds of expected rank is loose since we know nothing about its rank distribution without looking through the possible worlds. Here, we use sampling to extract some information on its rank distribution.

Suppose the set of all possible worlds based on partition P_i is $\mathcal{W}_i$. We sample m'_i elements from $\mathcal{W}_i$, the set of which is denoted as $\mathcal{W}'_i$. For site $s \in P_i$, its local rank in the sampled possible world is known. For all other possible worlds, we still use 0 and $n_i - 1$ as the lower and upper bounds. Then, we have the estimation for the local expected rank

$$ler^-_{P_i}(s) = \sum_{W \in \mathcal{W}'} Pr(W) r^{P_i}_W(s)$$

$$ler^+_{P_i}(s) = ler^-_{P_i}(s) + (n_i - 1)\left(1 - \sum_{W \in \mathcal{W}'} Pr(W)\right).$$

By sampling more possible worlds, the $\sum_{W \in \mathcal{W}'} Pr(W)$ part will be raised; hence, the estimation is more accurate. But increasing sampling rate also has an adverse impact on the performance that it will incur extra cost to examine more possible worlds. We will investigate the effect of sampling rate in the experiments later.

8 Experiments

In this section, we present results of our empirical study to verify the effectiveness and efficiency of the proposed techniques in this paper. In the following experiments, we use the data of the *Digital Chart of the World* from the R-tree-Portal [32]. We take one data set which consists of 9,203 cultural landmarks in North America as the sites. The other data set consisting 24,493 populated places in North America is employed to represent the centers of uncertain objects, whose uncertain regions are circles with radius r. Within each uncertain region, we generate m instances which follow three popular distributions *Uniform*, *Constrained Normal* with $\sigma = 0.3 \times r$ and *Zipf* with $z = 0.5$. The sites and the uncertain regions of objects are indexed by R-tree with default settings. For each workload, we issue 100 queries and measure their performance by average. All algorithms are implemented in Java and run on a PC with Intel P4 2.4 GHz and 1 GB memory. Table 3 summarizes the parameters and their default values used in the experiments.

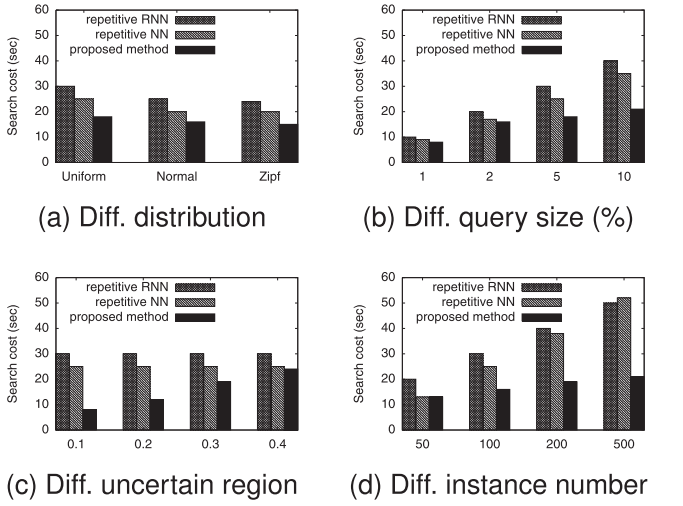


Fig. 6. Efficiency of PRNN search.

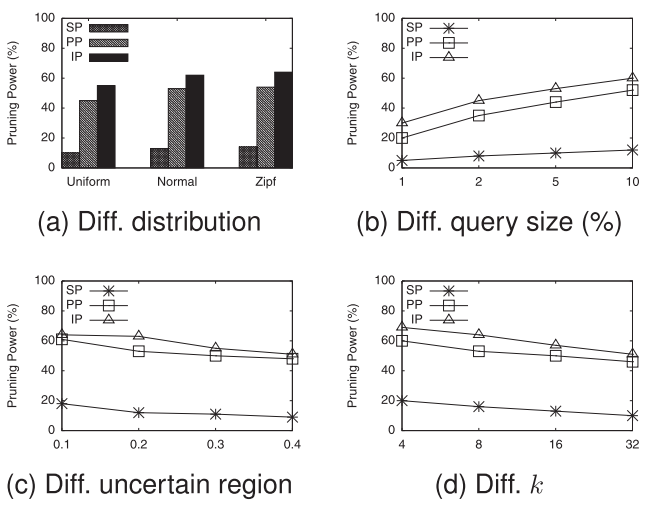


Fig. 7. Effectiveness of pruning.

8.1 PRNN Search Performance

In this section, we compare the proposed optimizations for the PRNN search step with two baseline approaches. The first one, called *repetitive RNN*, organizes all the instances of object set with a single R-tree and issues the traditional RNN query for each site inside the query region. The second method, called *repetitive NN*, indexes all the sites with an R-tree. For each uncertain object within the query region, it performs a batch NN search for the instances of the object to identify the sites that may influence this object.

Fig. 6a shows the results of the first set of experiments. It is easy to observe that our proposed approach achieve better performance on all data sets with different probability distributions. We also notice that the repetitive NN method runs faster than the repetitive RNN method, which is due to the fact that the RNN queries are more expensive than NN queries. Besides, all three algorithms run faster on normal and Zipf distributions than on uniform distribution, though the impact of distribution is not very significant.

Then, we compare their performance for different sizes of query region, the results of which are shown in Fig. 6b. Not surprisingly, all approaches need more time to finish for a larger query region, since more candidate sites and objects are involved with the query. However, the performance deterioration speed of the proposed approach is slower than the baseline methods. This is because they need to issue more RNN or NN queries as the query region expands, while the voronoi diagram only needs to be constructed once regardless the size of the query region.

The third experiment shows the relationship between the search costs and the uncertain region of objects. From Fig. 6c, we can see that the baseline approaches are barely affected by this factor, since they are only affected by the number of instances or sites. However, when the uncertain region expands, each object is more likely to be influenced by more sites, which means the proposed algorithm tends to traverse more deeply into the instance R-tree and hence costs more time.

At last, we examine how performance of each algorithm varies when the uncertain objects have different number of instances. As shown in Fig. 6d, the search costs of baseline approaches increases quickly when more instances are generated for each object. More interestingly, when the number of instances gets even larger, the repetitive NN method runs slower than the repetitive RNN method. The reason lies in that, the cost of the repetitive NN method is proportional to the number of NN queries issued which is equivalent to the number of instances within the query region. So, the performance of the repetitive NN method is more sensitive to this parameter. For the proposed approach, on the other hand, increasing instances while fixing the uncertain region just affects the density of the nodes of instance R-tree, which has no obvious effect on the search performance. Higher cost is caused by the worst case scenario, i.e., the instances within a leaf node are probed.

8.2 Effectiveness of Pruning

In this section, we compare the pruning effects of simple pruning method with the probabilistic pruning as well as the iterative pruning against different settings. In our experiments, the *pruning power* is measured by the ratio between the number of pruned sites and the number of candidate sites.

First, we test the pruning power on the data sets with different distributions, the results of which are shown in Fig. 7a. As expected, the simple pruning can only eliminate about 10 percent of the candidates, which is much worse than the other two methods, due to its strict comparison criteria. The iterative pruning method always has the best performance. Moreover, we observe that the pruning effects are better are skewed distribution than uniform distribution.

Next, we investigate the impact of query size on the pruning effects. From Fig. 7b, we can see that the pruning power of all methods increase as the query region grows. However, compared to the significant improvement of both *PP* and *IP*, the change on the effect of simple pruning method is neglectable.

Fig. 7c shows the results of pruning power with different uncertain regions. As we can see, the effects of all pruning methods are degrading with the increasing uncertain regions. This is because an object will be influenced by more sites as its uncertain region expands, which means the

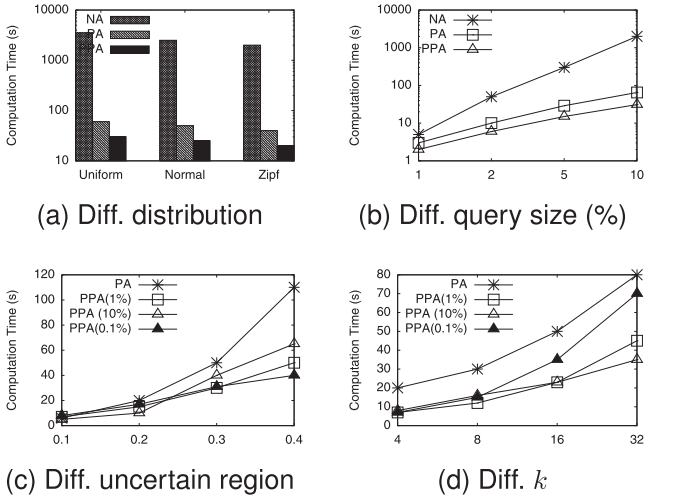


Fig. 8. Efficiency of rank evaluation.

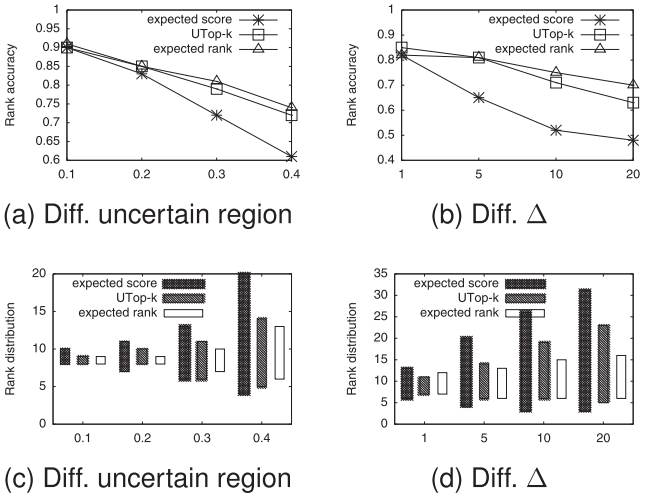


Fig. 9. Results quality analysis.

influence of a site may span a broader range of values, making it more difficult to get pruned.

In the fourth set of experiments, we study the impact of k on the pruning effects. The results are shown in Fig. 7d, in which the power of all pruning methods degrade as more results are required to return. This is expected since, with larger k, it is more difficult for a site to satisfy the pruning lemma.

8.3 Efficiency of Rank Evaluation

In this section, we compare the efficiency of different rank evaluation methods, namely naive algorithm (*NA*), partition-based algorithm (*PA*) and partition-pruning-based algorithm (*PPA*). In the following experiments, the candidate sites and objects are selected by the iterative pruning method.

First, we run all the algorithms on the data sets with different distributions and show their computation time in Fig. 8a. As we can see, by breaking the candidate sites into independent partitions, *PA* outperforms *NA* by about two orders of magnitude. Besides, *PPA* further reduce the computation time of *PA* by about 50 percent. We also notice that more time is needed on uniform distribution for all algorithms, which is caused by the worse pruning effects.

Fig. 8b illustrates the impact of query size on the efficiency of algorithms. When the query size is small, all the algorithms are still comparable since there are not many candidates. But the computation time of *NA* increases exponentially with the query size. On the other hand, the performance deterioration speed of both *PA* and *PPA* is much slower.

In the last two sets of experiments, we study the impact of uncertain region and k on the efficiency of algorithms, respectively. We omit the results of naive algorithm and show the performance of *PPA* with different sampling rate. According to Fig. 8c, when the uncertain region is small, all algorithms are comparable since there are only a few possible worlds to be checked in each partition. However, *PPA* methods gradually show advantage over *PA* as the uncertain region expands. Moreover, *PPA* with higher sampling rate is faster with the uncertain region because it estimates the expected ranks more accurate. But it also incurs extra cost which causes the performance worse than the one with lower sampling rate when uncertain region becomes large. From Fig. 8d, we can see that all algorithms need more time to complete as k increases. The performance of *PPA* with lower sampling rate is better when k is small, since even a loose estimation of the expected rank can still prune some sites. However, when k increases, its running time gradually approaches the *PA* because it is more difficult to prune by partial expected ranks.

8.4 Quality Analysis of Query Results

Due to the lack of a public available data set to examine the quality of different uncertain top-k query semantics, we design the following method to test the result quality. The ground truth is obtained by computing the influences in the original data set, i.e., the one with deterministic objects and sites. The weight of each object is randomly set to one of the four values $(\Delta^0, \ldots, \Delta^3)$, where Δ is a parameter and will be tuned in our experiments. Specifically, we quantify the quality of ranking results by two measures. The *rank accuracy* is measured by the fraction of the correct top-k results with respect to the ground truth. The *rank distribution* is illustrated by a bar in the following figures with high and low points, representing the worst rank of false negative and best rank of false positive respectively. So, a long bar stands for worse ranking quality than a short bar. We also include two other semantics, expected score and UTop-k [20], for comparison purpose.

First, we investigate the impact of the size of the uncertain region, the results of which are shown in Figs. 9a and 9c. It is observed that all query semantics can achieve high results quality, i.e., high accuracy and similar ranks in between false positives and false negatives, when the uncertain region is very small. This is because, most instances of each uncertain object are still influenced by the same sites as the original data set. In such a case, the choice of top-k semantics does not have obvious impact on the results quality. However, as the uncertain region expands, the instances of an uncertain object become more spread and may be influenced by more sites. As a result, the ranking accuracy of expected score drops quickly and ranks of false negatives may also be ranked far away from its correct position. But, we can notice that UTop-k and

expected rank can still deliver a relative good results since they both adopt the possible world semantics.

Then, we vary Δ to see how the way of assigning the weights affect the ranking quality. When $\Delta = 1$, i.e., all the objects have equal weights, all the top-k semantics have almost the same quality. As Δ increases, which means the gap between the weights of different objects becomes greater, the advantage of expected rank semantic gets more significant. The reason is that both expected score and UTop-k depends on the values, which are not as stable as the relative ranks. Especially, when the differences among the weights are great, an influence can be more easily biased by an object with a large weight even it is in a very unlikely possible world.

9 Conclusion and Future Work

This paper studies a novel query on uncertain databases, namely *uncertain top-k influential site query*. We formally define this query based on the intuitive expected rank semantics. We also propose pruning techniques and partition-based algorithms to improve the querying performance. Experimental results verifies the effectiveness and efficiency of the techniques in this paper.

There are several challenges that we plan to address in the future. One of them is to apply other uncertain top-k semantics as reviewed in Section 2 and design efficient processing algorithms. Another interesting problem is to consider uncertainty in both sites and objects, which requires more sophisticated algorithms. Finally, we plan to extend our solution to handle tuple uncertainty as well.

Acknowledgments

The authors wish to thank the anonymous reviewers for the comments and suggestions that have greatly improved the paper. This work was supported by the NSFC grant 60925008, and ARC grants DP110103423 and DP0987557.

References

[1] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic Skylines on Uncertain Data," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 15-26, 2007.

[2] J. Chen and R. Cheng, "Efficient Evaluation of Imprecise Location-Dependent Queries," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 586-595, 2007.

[3] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Evaluatizng Probabilistic Queries over Imprecise Data," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD),* pp. 551-562, 2003.

[4] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. Vitter, "Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 876-887, 2004.

[5] Y. Tao, R. Cheng, X. Xiao, W. Ngai, B. Kao, and S. Prabhakar, "Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 922-933, 2005.

[6] R. Cheng, J. Chen, M. Mokbel, and C. Chow, "Probabilistic Verifiers: Evaluating Constrained Nearest-Neighbor Queries over Uncertain Data," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 973-982, 2008.

[7] R. Cheng, L. Chen, J. Chen, and X. Xie, "Evaluating Probability Threshold K-Nearest-Neighbor Queries over Uncertain Data," *Proc. Int'l Conf. Extending Database Technology: Advances in Database Technology (EDBT),* pp. 672-683, 2009.

[8] H. Kriegel, P. Kunath, and M. Renz, "Probabilistic Nearest-Neighbor Query on Uncertain Objects," *Proc. 12th Int'l Conf. Database Systems for Advanced Applications (DASFAA),* 2007.

[9] X. Lian and L. Chen, "Efficient Processing of Probabilistic Reverse Nearest Neighbor Queries over Uncertain Data," *VLDB J.,* vol. 18, no. 3, pp. 787-808, 2009.

[10] X. Lian and L. Chen, "Monochromatic and Bichromatic Reverse Skyline Search over Uncertain Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD),* pp. 213-226, 2008.

[11] H. Kriegel, P. Kunath, M. Pfeifle, and M. Renz, "Probabilistic Similarity Join on Uncertain Data," *Proc. 11th Int'l Conf. Database Systems for Advanced Applications (DASFAA),* p. 295, 2006.

[12] V. Ljosa and A. Singh, "Top-K Spatial Joins of Probabilistic Objects," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 566-575, 2008.

[13] T. Xia, D. Zhang, E. Kanoulas, and Y. Du, "On Computing Top-T Most Influential Spatial Sites," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* p. 957, 2005.

[14] F. Korn and S. Muthukrishnan, "Influence Sets Based on Reverse Nearest Neighbor Queries," *ACM SIGMOD Record,* vol. 29, no. 2, pp. 201-212, 2000.

[15] J. Kang, M. Mokbel, S. Shekhar, T. Xia, and D. Zhang, "Continuous Evaluation of Monochromatic and Bichromatic Reverse Nearest Neighbors," *Proc. Int'l Conf. Data Eng. (ICDE),* 2007.

[16] Y. Tao, D. Papadias, and X. Lian, "Reverse KNN Search in Arbitrary Dimensionality," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* p. 755, 2004.

[17] G. Cormode, F. Li, and K. Yi, "Semantics of Ranking Queries for Probabilistic Data and Expected Ranks," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 305-316, 2009.

[18] T. Ge, S. Zdonik, and S. Madden, "Top-K Queries on Uncertain Data: On Score Distribution and Typical Answers," *Proc. 35th ACM SIGMOD Int'l Conf. Management of Data (SIGMOD),* pp. 375-388, 2009.

[19] C. Re, N. Dalvi, and D. Suciu, "Efficient Top-K Query Evaluation on Probabilistic Data," *Proc. Int'l Conf. Data Eng. (ICDE),* 2007.

[20] M. Soliman, I. Ilyas, and K. Chang, "Top-K Query Processing in Uncertain Databases," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 896-905, 2007.

[21] K. Yi, F. Li, D. Srivastava, and G. Kollios, "Efficient Processing of Top-K Queries in Uncertain Databases with X-Relations," *IEEE Trans. Knowledge and Data Eng.,* vol. 20, no. 12, pp. 1669-1682, Dec. 2008.

[22] X. Zhang and J. Chomicki, "Semantics and Evaluation of Top-K Queries in Probabilistic Databases," *Proc. Int'l Workshop Database Ranking (DBRank),* 2008.

[23] Y. Zhang, X. Lin, G. Zhu, W. Zhang, and Q. Lin, "Efficient Rank Based Knn Query Processing over Uncertain Data," *Proc. Int'l Conf. Data Eng. (ICDE),* 2010.

[24] X. Lian and L. Chen, "Probabilistic Group Nearest Neighbor Queries in Uncertain Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 20, no. 6, pp. 809-824, June 2008.

[25] M. Hua, J. Pei, W. Zhang, and X. Lin, "Ranking Queries on Uncertain Data: A Probabilistic Threshold Approach," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD),* pp. 673-686, 2008.

[26] J. Li, B. Saha, and A. Deshpande, "A Unified Approach to Ranking in Probabilistic Databases," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 769-780, 2009.

[27] C. Yang and K. Lin, "An Index Structure for Efficient Reverse Nearest Neighbor Queries," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 485-492, 2001.

[28] I. Stanoi, D. Agrawal, and A. Abbadi, "Reverse Nearest Neighbor Queries for Dynamic Databases," *Proc. ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery,* pp. 44-53, 2000.

[29] Y. Tao, M. Yiu, and N. Mamoulis, "Reverse Nearest Neighbor Search in Metric Spaces," *IEEE Trans. Knowledge and Data Eng.,* vol. 18, no. 9, pp. 1239-1252, Sept. 2006.

[30] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms.* The MIT Press, 1990.

[31] W. Hoeffding, "Probability Inequalities for Sums of Bounded Random Variables," *J. Am. Statistical Assoc.,* vol. 58, no. 301, pp. 13-30, 1963.

[32] Y. Theodoridis, "The R-Tree-Portal," http://www.rtreeportal.org, 2003.

Kai Zheng received the MS degree in 2009 from Fudan University of China. He is currently working toward the PhD degree at the University of Queensland. His research interests include spatial database and uncertain database while focusing on designing efficient query processing algorithms and effective indexing structures.

Zi Huang received the PhD degree in computer science from The University of Queensland, Australia, in 2007. Currently, she is a research fellow with the DKE group, the University of Queensland. Her research interests include multimedia information retrieval, multimedia database management, indexing and query processing, and bioinformatics. She is a member of the IEEE.

Aoying Zhou received the PhD degree from Fudan University in 1993. Currently, he is a professor in Software Engineering Institute at East Normal University of China. His research interests include web search and mining, data stream management, uncertain data management, distributed storage and computing, and web service. He is a member of the IEEE.

Xiaofang Zhou received the BSc and MSc degrees in computer science from Nanjing University, China, in 1984 and 1987, respectively, and the PhD degree in computer science from The University of Queensland, Australia, in 1994. He is a professor of computer science at The University of Queensland. His research interests include spatial and multimedia databases, high performance query processing, web information systems, data mining, bioinformatics, and e-research.