Probabilistic Reverse Nearest Neighbor Queries on Uncertain Data

Muhammad Aamir Cheema, Xuemin Lin, Wei Wang, Wenjie Zhang and Jian Pei, *Senior Member, IEEE*

Abstract—Uncertain data is inherent in various important applications and reverse nearest neighbor (RNN) query is an important query type for many applications. While many different types of queries have been studied on uncertain data, there is no previous work on answering RNN queries on uncertain data. In this paper, we formalize probabilistic reverse nearest neighbor 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. We develop an efficient algorithm based on various novel pruning approaches that solves the probabilistic RNN queries on multidimensional uncertain data. The experimental results demonstrate that our algorithm is even more efficient than a sampling-based approximate algorithm for most of the cases and is highly scalable.

Index Terms—Query Processing, Reverse Nearest Neighbor Queries, Uncertain Data, Spatial Data.

1 Introduction

Given a set of data points P and a query point q, a reverse nearest neighbor query is to find every point $p \in P$ such that $dist(p,q) \leq dist(p,p')$ for every $p' \in (P-p)$. In this paper, we formalize and study probabilistic RNN query that is to find the probable reverse nearest neighbors on uncertain data with probability higher than a given threshold.

Uncertain data is inherent in many important applications such as sensor databases, moving object databases, market analysis, and quantitative economic research. In these applications, the exact values of data might be unknown due to limitation of measuring equipment, delayed data updates, incompleteness, or data anonymization to preserve privacy.

Usually an uncertain object is represented in two ways: 1) using a probability density function [4], [6] (continuous case) and 2) using all possible instances [22], [17] each with an assigned probability (discrete case). In this paper, we investigate discrete cases.

Probabilistic RNN queries have many applications. Consider the example in Fig. 1, where three residential blocks A, B and Q are shown. The houses within each block are shown as small circles. The centroid of each residential block is shown as a hollow triangle. For privacy reasons, we may only know the residential blocks in which the people live (or zipcode) but we do not have any information about the exact addresses of their houses. We can assign some probability to each possible location of a person in his residential block. e.g; the exact location of a person living in A is a_1 with 0.5 probability.

Conventional queries on these residential blocks may use distance functions like the distance between the centroids of two blocks. However, the results provided by the conventional queries may not be meaningful. There are two major limitations for conventional queries on such data[1].

1) The conventional queries do not consider the locations of houses within each residential block. This affects quality of the reported results. For instance, if the distance between centroids of two residential blocks is used as distance function, the closest block of A is B (in other words the person living in A is *not* the RNN of someone living in Q). However, if the locations of houses within each block are considered, we find that for most of the houses in A, the houses in Q are closer than the houses in B. For example, the distance of a_1 to every house in Q is less that its distance to any house in B. Similarly, the distance of a_2 to every house in Q is less than its distance to b_1. Which means, a person living in A has high chances to be the RNN of some person living in Q.

2) Conventional queries do not report the probability of objects to be the answer (an object is either a RNN or not a RNN). On the other hand, probabilistic reverse nearest neighbor queries provide more information by including the probability of an object to be the answer. For example, a probabilistic reverse nearest neighbor query reports that the probability of a person living in block A to become the RNN of a person living in Q is 0.75 according to the possible world semantics (see example 1). This type of results are more meaningful and interesting.

Probabilistic RNN queries have applications in privacy preserving location-based services where the exact location of every user is obfuscated into a cloaked spatial region [16]. However, the users might still be interested in finding their reverse nearest neighbors. We can model this problem to finding probabilistic reverse nearest neighbor by assigning confidence level to some possible locations of every user within his/her respective cloaked spatial region. Probabilistic RNN queries may also be useful to identify similar trading

- *Muhammad Aamir Cheema is with the School of Computer Science and Engineering, University of New South Wales, Australia. E-mail: macheema@cse.unsw.edu.au*
- *Xuemin Lin, Wei Wang and Wenjie Zhang are with the University of New South Wales and NICTA. E-mails: {lxue,weiw,zhangw}@cse.unsw.edu.au*
- *Jian Pei is with Simon Fraser University, Canada. E-mail: jpei@cs.sfu.ca*

1. Other distance functions like maximum distance, minimum distance and aggregated distance also have these limitations.

trends in stock markets. Each stock has many deals. A deal (transaction) is recorded by the price (per share) and the volume (number of shares). For a given stock s, clients may be interested in finding all other stocks that have trading trends more similar to s than others. In such application, we can treat each stock as an uncertain object and its deals as its uncertain instances. There are a number of other applications for the queries that consider the proximity of uncertain objects [4], [6], [14] and the applications of RNNs on uncertain objects are very similar.

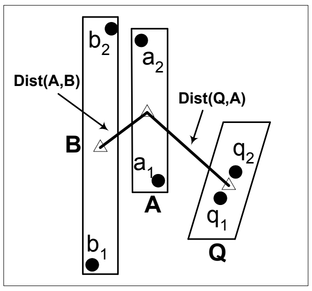


Fig. 1: An example of a probabilistic RNN query

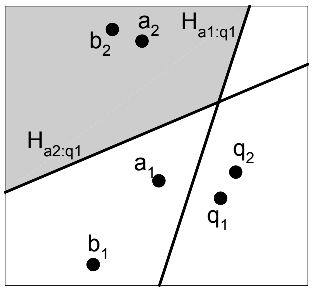


Fig. 2: Any point in shaded area cannot be RNN of q

Probabilistic RNN query processing poses new challenges in designing new efficient algorithms. Although RNN query processing has been extensively studied based on various pruning methods, these pruning techniques either cannot be directly applied to probabilistic RNN queries or become inefficient. For example, the perpendicular bisectors adopted in the state-of-the-art RNN query processing algorithm [20] assume that objects are spatial points. In contrast, uncertain objects have arbitrary shapes of their uncertain regions. In addition, applying the pruning rules on the instance level of uncertain objects is extremely expensive as each uncertain object usually has a large number of instances.

Another unique challenge in probabilistic RNN queries is that the verification of candidate objects usually incurs substantial cost due to large number of instances in each uncertain object. By verification, we mean computing the exact probability of an object being the RNN of the query and testing whether it qualifies the probabilistic threshold or not. Note that instances from objects that are close to the candidate objects also need to be considered in the verification phase.

In this paper, we formalize the problem of probabilistic RNN queries on uncertain data using the semantics of *possible worlds*. We present a new probabilistic RNN query processing framework that employs (i) several novel pruning approaches exploiting the probability threshold and geometric, topological and metric properties. (ii) a highly optimized verification method that is based on careful upper and lower bounding of the RNN probability of candidate objects.

Our contributions in this paper are as follows:

- To the best of our knowledge, we are the first to formalize the problem of probabilistic reverse nearest neighbors based on the possible worlds semantics.
- We develop efficient query processing algorithm of probabilistic RNN queries. The new method is based on non-trivial pruning rules especially designed for uncertain data and the probability threshold. Although in this paper, we focus on *discrete* case where each object is represented by some possible probable instances, our pruning rules can be applied to the *continuous* case where each uncertain object is represented by a probability density function.
- To better understand performance of our proposed approach, we devise a baseline exact algorithm and a sampling-based approximate algorithm. Experiment results on synthetic and real datasets show that our algorithm is much more efficient than the baseline algorithm and performs better than the approximate algorithm for most of the cases and is scalable.

The rest of the paper is organized as follows: In Section 2, we formalize the problem and present the preliminaries and notations used in this paper. Our proposed pruning rules are presented in Section 3. Section 4 presents our proposed algorithm for answering probabilistic reverse nearest neighbor queries. Section 5 evaluates the proposed methods with extensive experiments and the related work is presented in Section 6. Section 7 concludes the paper.

2 Problem Definition and Preliminaries

2.1 Problem Definition

Given a set of data points P and a query point q, a conventional reverse nearest neighbor query is to find every point $p \in P$ such that $dist(p,q) \leq dist(p,p')$ for every $p' \in (P-p)$.

Now we define probabilistic reverse nearest neighbor queries. Consider a set of *uncertain objects* $\mathcal{U} = \{U_1, ..., U_n\}$. Each uncertain object U_i consists of a set of *instances* $\{u_1, ..., u_m\}$. Each instance u_j is associated with a probability p_{u_j} called *appearance probability* with the constraint that $\sum_{j=1}^{m} p_{u_j} = 1$. We assume that the probability of each instance is independent of other instances. A *possible world* $W = \{u_1, ..., u_n\}$ is a set of instances with one instance from each uncertain object. The probability of W to appear is $P(W) = \prod_{i=1}^{n} p_{u_i}$. Let Ω be the set of all possible worlds, then $\sum_{W \in \Omega} P(W) = 1$.

The probability $RNN_Q(U_i)$ of any uncertain object U_i to be the RNN of an uncertain object Q in all possible worlds can be computed as;

$$RNN_Q(U_i) = \sum_{(u,q), u \in U_i, q \in Q} p_q \cdot p_u \cdot RNN_q(u) \qquad (1)$$

$RNN_q(u)$ is the probability that an instance $u \in U_i$ is the RNN of an instance $q \in Q$ in any possible world W given that both u and q appear in W.

$$RNN_q(u) = \prod_{V \in (\mathcal{U} - U_i - Q)} (1 - \sum_{v \in V, dist(u,v) < dist(u,q)} p_v) \quad (2)$$

Given a set of uncertain objects $\mathcal{U}$ and a probability threshold ρ, problem of finding probabilistic reverse nearest neighbors of any uncertain object Q is to find every uncertain object $U_i \in \mathcal{U}$ such that $RNN_Q(U_i) \geq \rho$.

Example 1: Consider the example of Fig. 1 where the uncertain objects A, B and Q are shown. Assume that the appearance probability of each instance is 0.5. According to Equation (2), $RNN_{q_1}(a_1) = 1$ because a_1 is closer to q_1 than it is to b_1 or b_2. Also $RNN_{q_1}(a_2) = 1 - 0.5$ because $dist(a_2, b_2) < dist(a_2, q_1)$. Note that b_1 does not affect the probability of a_2 to be the RNN of q_1 because $dist(a_2, b_1) > dist(a_2, q_1)$. Similarly, $RNN_{q_2}(a_1) = 1$ and $RNN_{q_2}(a_2) = 0.5$. According to Equation (1), $RNN_Q(A) = (0.5 \times 0.5 \times 1) + (0.5 \times 0.5 \times 1) + (0.5 \times 0.5 \times 0.5) + (0.5 \times 0.5 \times 0.5) = 0.75$. RNN probability of B can be computed similarly and $RNN_Q(B) = 0.25$. If the probability threshold ρ is 0.7, then the object A is reported as result. ■

2.2 Preliminaries

The filter-and-refine paradigm is widely adopted in processing RNN queries in spatial databases. The idea is to quickly prune away points which are closer to another point (usually called *filtering point*) than to the query point. The state-of-the-art pruning rule is based on perpendicular bisector [20]. It consists of two phases: the pruning phase and the verification phase.

Hence, some objects are used to filter other objects and are called *filtering objects*. Objects that cannot be filtered are called *candidate objects*. The pruning in RNN query processing involves three objects, the query, the filtering object and a candidate object. We use R_Q, R_{fil} and R_{cnd} to denote the smallest hyper-rectangles enclosing uncertain query object, filtering object and candidate object, respectively.

Table 1 defines the symbols and notations used throughout this paper.

TABLE 1: Notations

Notation	Definition
U	an uncertain object
u_i	i^{th} instance of uncertain object U
$B_{x:q}$	a perpendicular bisector between point x and q
$H_{x:q}$	a half-space defined by $B_{x:q}$ containing point x
$H_{q:x}$	a half-space defined by $B_{x:q}$ containing point q
$H_{a:b} \cap H_{c:d}$	intersection of the two half-spaces
$P[i]$	value of point P in the i^{th} dimension
R_U	minimum bounding rectangle (MBR) enclosing all instances of an uncertain object U

3 Pruning Rules

Although the pruning for RNN query processing in spatial databases has been well studied, it is *non-trivial* to devise pruning strategies for RNN query processing on uncertain data. For example, if we naïvely use every instance of a filtering object to perform bisector pruning [20], it will incur a huge computation cost due to large number of instances in each uncertain object. Instead, we devise non-trivial generalization of bisector pruning for minimum bounding rectangles (MBRs) of uncertain objects based on a novel notion of *normalized half-space*.

Verification is extremely expensive in probabilistic RNN query processing because, in order to verify an object as probabilistic RNN, we need to take into consideration not only the instances of this object but also the instances of query object and other nearby objects. Hence it is important to devise efficient pruning rules to reduce the number of objects that need verification. In this section, we present several pruning rules from the following orthogonal perspectives:

- Half-space based pruning that exploits geometrical properties (Section 3.1)
- Dominance based pruning that exploits topological properties (Section 3.2)
- Metric based pruning (Section 3.3)
- Probabilistic pruning that exploits the probability threshold (Section 3.4)

3.1 Half-space Pruning

Consider a query point q and a filtering object U that has n instances $\{u_1, u_2, \ldots, u_n\}$. Let $H_{u_i:q}$ be the half-space between q and u_i. Any instance $u \notin U$ that lies in $\cap_{i=1}^{n} H_{u_i:q}$ has zero probability to be the RNN of q because by the property of $H_{u_i:q}$, u is closer to every u_i than to q.

Example 2: Consider the example of Fig. 2 where the bisectors between q_1 and the instances of A are drawn and the half-spaces $H_{a_1:q_1}$ and $H_{a_2:q_1}$ are shown. Intersection of the two half-spaces is shown shaded and any point that lies in the shaded area is closer to both a_1 and a_2 than q_1. For this reason, b_2 cannot be the RNN of q_1 in any possible world. ■

This pruning is very expensive because we need to compute intersection of all half-spaces $H_{u_i:q}$ for every $u_i \in U$. Below we present our pruning rules that utilize the MBR of the entire filtering object, R_{fil}, to prune the candidate object with respect to a query instance q or the MBR of uncertain query object Q.

3.1.1 *Pruning using R_{fil} and an instance q*

First we present the intuition. Consider the example of Fig. 3 where we know that the point p lies on a line MN but we do not know the exact location of p on this line. The bisectors between q and the end points of the line (M and N) can be used to prune the area safely. In other words, any point that lies in the intersection of half-spaces $H_{M:q}$ and $H_{N:q}$ (grey area) can never be the RNN of q. It can be proved that whatever be the location of point p on the line MN, the half-space $H_{p:q}$ always contains $H_{M:q} \cap H_{N:q}$. Hence any point p' that lies in $H_{M:q} \cap H_{N:q}$ would always be closer to p than to q and for this reason cannot be the RNN of q.

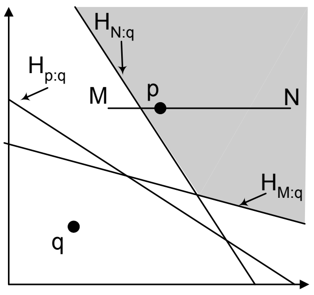


Fig. 3: The exact location of the point p on line MN is not known

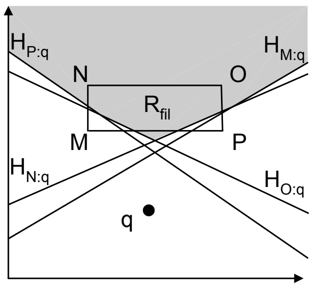


Fig. 4: Any point in shaded area cannot be RNN of q in any possible world

Based on the above observation, below we present a pruning rule for the case when the exact location of a point p is unknown within some hyper-rectangle R_{fil}.

PRUNING RULE 1 : Let R_{fil} be a hyper-rectangle and q be a query point. For any point p that lies in $\bigcap_{i=1}^{2^d} H_{C_i:q}$ (C_i is the i^{th} corner of R_{fil}), $dist(p,q) > maxdist(p, R_{fil})$ and thus p cannot be the RNN of q.

The pruning rule is based on Lemma 4 that is proved in our technical report [5].

Consider the example of Fig. 4. Any point that lies in shaded area is closer to every point in rectangle R_{fil} than to q. Note that if R_{fil} is a hyper rectangle that encloses *all* instances of the filtering object U_i then any instance $u \in U_{j,j \neq i}$ that lies in $\bigcap_{i=1}^{2^d} H_{C_i:q}$ can never be the RNN of q in any possible world.

3.1.2 Pruning using R_{fil} and R_Q

Pruning rule 1 prunes the area such that any point lying in it can never be the RNN of some instance q. However, the points in the pruned area may still be the RNNs of other instances of the query. Now, we present a pruning rule that prunes the area using R_{fil} and R_Q such that any point that lies in the pruned area cannot be the RNN of *any* instance of Q.

Consider the example of Fig. 5 where the exact location of the query point q on line MN is not known. Unfortunately, in contrast to the previous case of Fig. 3, the bisectors between p and the end points of the line MN do *not* define the area that can be pruned. If we prune the area $H_{p:M} \cap H_{p:N}$ (the grey area), we may miss some point p' that is the RNN of q. Fig. 5 shows a point p' that is the RNN of q but lies in the shaded area. This is because the half-space $H_{p:q}$ does not contain $H_{p:M} \cap H_{p:N}$. This makes the pruning using R_{fil} and R_Q challenging.

Note that if $H_{p:N}$ is moved such that it passes through the point where $H_{p:q}$ intersects $H_{p:M}$ then $H_{p:M} \cap H_{p:N}$ would be contained by $H_{p:q}$. We note that in the worst case when p lies infinitesimally close to point M, $H_{p:q}$ and $H_{p:M}$ intersect each other at point c which is the centre of line joining p and M. Hence, in order to safely prune the area, the half-space $H_{p:N}$ should be moved such that it passes through the point c. The point c is shown in Fig. 5. A half-space that is moved to the point c is called a *normalized* half-space and a half-space $H_{p:N}$ that is normalized is denoted as $H'_{p:N}$. Fig. 5 shows $H'_{p:N}$ in broken line and $H'_{p:N} \cap H_{p:M}$ (the dotted shaded area) can be safely pruned.

The correctness proof of the above observation is lengthy though it is quite intuitive. Thus we omit it from the paper. The interested readers may read the proof in our technical report [5] for a more general case when both the query and data objects are represented by hyper-rectangles in d dimensional space (Lemma 5). Before we present our pruning rule for the general case that uses 2^d half-spaces to prune the area using hyper-rectangles R_Q and R_{fil}, we define the following concepts:

Antipodal Corners: Let C be a corner of rectangle $R1$ and C' be a corner in $R2$, the two corners are called *antipodal*

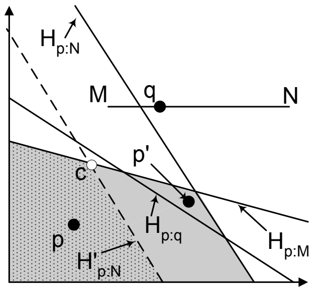


Fig. 5: Any point in dotted area can never be RNN of q

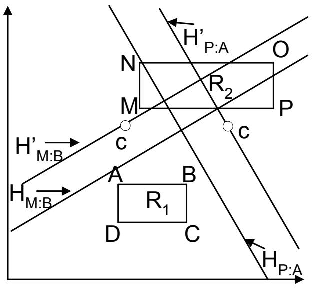


Fig. 6: Antipodal corners and normalized half-spaces

corners[2] if for every dimension i where $C[i] = R1_L[i]$ then $C'[i] = R2_H[i]$ and for every dimension j where $C[j] = R1_H[j]$ then $C'[j] = R2_L[j]$. Fig. 6 shows two rectangles $R1$ and $R2$. The corners D and O are antipodal corners. Similarly, other pairs of antipodal corners are (B, M), (C, N) and (A, P).

Antipodal Half-Space: A half-space that is defined by the bisector between two antipodal corners is called *antipodal half-space*. Fig. 6 shows two antipodal half-spaces $H_{M:B}$ and $H_{P:A}$.

Normalized Half-Space: Let B and M be two points in hyper-rectangles $R1$ and $R2$, respectively. The normalized half-space $H'_{M:B}$ is a space defined by the bisector between M and B that passes through a point c such that $c[i] = (R1_L[i] + R2_L[i])/2$ for all dimensions i for which $B[i] > M[i]$ and $c[j] = (R1_H[i] + R2_H[j])/2$ for all dimensions j for which $B[j] \leq M[j]$. Fig. 6 shows two normalized (antipodal) half-spaces $H'_{M:B}$ and $H'_{P:A}$. The point c for each half-space is also shown. The inequalities (3) and (4) define the half-space $H_{M:B}$ and its normalized half-space $H'_{M:B}$, respectively.

$$\sum_{i=1}^{d}(B[i]-M[i]) \cdot x[i] < \sum_{i=1}^{d} \frac{(B[i]-M[i])(B[i]+M[i])}{2} \quad (3)$$

$$\sum_{i=1}^{d}(B[i]-M[i]) \cdot x[i] < \sum_{i=1}^{d}(B[i]-M[i]) \times \begin{cases} \dfrac{(R1_L[i]+R2_L[i])}{2}, & \text{if } B[i] > M[i] \\ \dfrac{(R1_H[i]+R2_H[i])}{2}, & \text{otherwise} \end{cases} \quad (4)$$

Note that the right hand side of the Equation (3) cannot be smaller than the right hand side of Equation (4). For this reason $H'_{MB} \subseteq H_{MB}$.

Now, we present our pruning rule.

PRUNING RULE 2 : Let R_Q and R_{fil} be two hyper-rectangles. For any point p that lies in $\bigcap_{i=1}^{2^d} H'_{C_i:C'_i}$, $mindist(p, R_Q) > maxdist(p, R_{fil})$ where $H'_{C_i:C'_i}$ is normalized half-space between C_i (the i^{th} corner of the rectangle R_{fil}) and its antipodal corner C'_i in R_Q.

The proof of correctness is non-trivial and can be found in Lemma 5 in our technical report [5].

2. $R_L[i]$ (resp. $R_H[i]$) is the lowest (resp. highest) coordinate of a hyper-rectangle R in i^{th} dimension

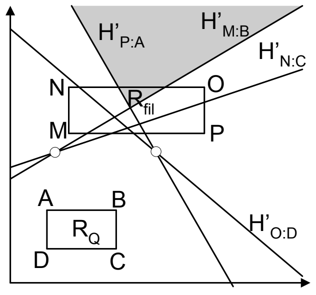


Fig. 7: Any point in shaded area can never be RNN of any $q \in Q$

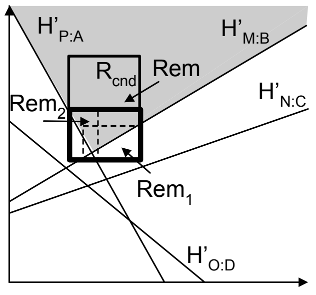


Fig. 8: Clipping part of the candidate object R_{cnd} that can not be pruned

Consider the example of Fig. 7 where the normalized antipodal half-spaces are drawn and their intersection is shown shaded. Any point that lies in the shaded area is closer to every point in rectangle R_{fil} than every point in rectangle R_Q.

Note that if R_{fil} and R_Q are the MBRs enclosing all instances of an uncertain object U_i and query object Q, respectively, any instance $u \in U_{j,j\neq i}$ that lies in the pruned region, $\bigcap_{i=1}^{2^d} H'_{C_i:C'_i}$, cannot be RNN of any instance of $q \in Q$ in any possible world. Even if the pruning region partially overlaps with R_{fil}, we can still trim the part of any other hyper-rectangle $R_{U_{j,j\neq i}}$ that falls in the pruned region. It is known that exact trimming becomes inefficient in high dimensional space, therefore, we adopt the loose trimming of R_{cnd} proposed in [20].

Algorithm 1 : hspace_pruning (Q, R_{fil}, R_{cnd})

Input: Q: an MBR containing instances of Q ; R_{fil}: the MBR to be used for trimming R_{cnd}: the candidate MBR to be trimmed
Description:

```
 1: $Rem = \varnothing$  // Remnant rectangle
 2: for each corner $C_i$ of $R_{fil}$ do
 3:     if $Q$ is a point then
 4:         $Rem_i$ = clip($R_{cnd}, H_{C_i:Q}$)  // clipping algorithm [10]
 5:     else if $Q$ is a hyper-rectangle then
 6:         $C'_i$ = antipodal corner of $C_i$ in $Q$
 7:         $Rem_i$ = clip($R_{cnd}, H'_{C_i:C'_i}$)  // clipping algorithm [10]
 8:     enlarge $Rem$ to enclose $Rem_i$
 9:     if $Rem = R_{cnd}$ then
10:         return $R_{cnd}$
11: return $Rem$
```

The overall half space pruning algorithm that integrates pruning rules 1 and 2 is illustrated in Algorithm 1. For each half-space, we use the clipping algorithm in [10] to find a *remnant* rectangle $Rem_i \subseteq R_{cnd}$ that cannot be pruned (lines 4 and 7). After all the half-spaces have been used for pruning, we calculate the MBR $Rem \subseteq R_{cnd}$ as the minimum bounding hyper rectangle covering every Rem_i. As such, we trim the original R_{cnd} to Rem.

For better illustration we zoom Fig. 7 and show the clipping of a hyper-rectangle R_{cnd} in Fig. 8. The algorithm returns Rem_1, Rem_2 (rectangles shown with broken lines) when $H'_{M:B}$ and $H'_{P:A}$ are parameters to the clipping algorithm, respectively. For the half-spaces $H'_{N:C}$ and $H'_{O:D}$ the whole hyper-rectangle R_{cnd} can be pruned so the algorithm returns ϕ. The remnant hyper-rectangle Rem is an MBR that encloses Rem_1 and Rem_2. Note that at any stage if the remnant rectangle Rem becomes equal to R_{cnd}, the clipping by other bisectors is not needed so R_{cnd} is returned without further clipping (line 10).

3.2 Dominance Pruning

We first give the intuition behind this pruning rule. Fig. 9 shows another example of pruning by using pruning rule 2 in two dimensional space. The normalized half-spaces are defined such that if R_{fil} is fully dominated[3] by R_Q in all dimensions then all the normalized antipodal half-spaces meet at point F_p as shown in Fig. 9. We also observe that for the case when R_{fil} is fully dominated by R_Q, the angle between the half-spaces that define the pruned area (shown in grey) is always greater than $90°$. Based on these observations, it can be verified that the space dominated by F_p (the dotted-shaded area) can be pruned [4].

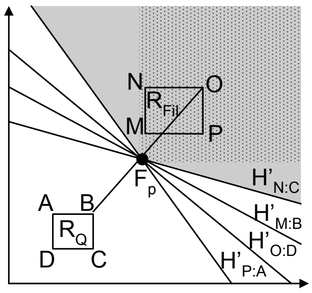


Fig. 9: Pruning area of half-space pruning and dominance pruning

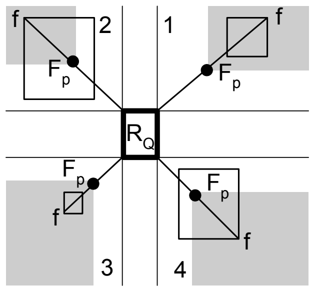


Fig. 10: Dominance Pruning: Shaded areas can be pruned

Let R_Q be the MBR containing instances of Q. We can obtain the 2^d regions as shown in Fig. 10. Let R_{U_i} be an MBR of a filtering object R_{fil} that lies completely in one of the 2^d regions. Let f be the furthest corner of R_{U_i} from R_Q and n be the nearest corner of R_Q from f. The *frontier point* F_p lies at the centre of line joining f and n.

PRUNING RULE 3 : Any instance $u \in U_j$ that is dominated by the frontier point F_p of a filtering object cannot be RNN of any $q \in Q$ in any possible world.

Fig. 10 shows four examples of dominance pruning (one in each region). In each partition the shaded area is dominated by F_p and can be pruned. Note that if R_{fil} is not fully dominated by R_Q, we cannot use this pruning rule because the normalized antipodal half-spaces in this case do not meet at the same point. For example, the four normalized antipodal half-spaces intersect at two points in Fig. 7. In general, the pruning power of this rule is less than that of the half-space pruning. Fig. 9 shows the area pruned by the half-space pruning (shaded area) and dominance pruning (dotted area).

The main advantage of this pruning rule is that the pruning procedure is computationally more efficient than the half-space pruning, as checking the dominance relationship and trimming the hyper-rectangles is easier.

3. If every point in R_1 is dominated (dominance relationship as defined in skylines) by every point in R_2 we say that R_1 is fully dominated by R_2.
4. Formal proof is given in Lemma 6 of our technical report [5]

3.3 Metric Based Pruning

PRUNING RULE 4 : An uncertain object R_{cnd} can be pruned if $maxdist(R_{cnd}, R_{fil}) < mindist(R_{cnd}, R_Q)$.

This pruning approach is the least expensive. Note that it cannot prune part of R_{cnd}, i.e., it either invalidates all the instances of R_{cnd} or does nothing.

3.4 Probabilistic Pruning

Note that we did not discuss probability threshold while presenting previous pruning rules. In this section, we present a pruning rule that exploits the probability threshold and embeds it in all previous pruning rules to increase their pruning powers.

A simple exploitation of the probability threshold is to trim the candidate object using previous pruning rules and then prune the object if the accumulative appearance probability of instances within its remnant rectangle is less than the threshold. Next, we present a more powerful pruning rule that is based on estimating an upper bound of the RNN probability of candidate objects.

First, we present an observation deduced from Lemma 5 in our technical report [5]. In previous pruning rules, we prune some area using MBR of a query object R_Q and a filtering object R_{fil}. We observe that the area pruned by using R'_Q and R'_{fil} always contains the area pruned by R_Q and R_{fil} where $R'_Q \subseteq R_Q$ and $R'_{fil} \subseteq R_{fil}$. Fig. 11 shows an example. The shaded area is pruned when R'_Q and R_{fil} are used for pruning and the dotted shaded area is pruned when R_Q and R_{fil} are used. Note that this observation also holds for the dominance pruning.

We can use the observation presented above to prune the objects that cannot have RNN probability greater than the threshold. First, we give a formal description of this pruning rule and then we give an example.

PRUNING RULE 5 : Let the instances of Q be divided into n disjoint[5] sets $\{Q_1, Q_2, ..., Q_n\}$ and R_{Q_i} be the minimum bounding rectangle enclosing *all* instances in Q_i. Let $\{R_{cnd_1}, R_{cnd_2}, ..., R_{cnd_n}\}$ be the set of bounding rectangles such that each R_{cnd_i} contains the instances of the candidate object that cannot be pruned for Q_i using any of the pruning rules. Let $P^{R_{Q_i}}$ and $P^{R_{cnd_i}}$ be the total appearance probabilities of instances in Q_i and R_{cnd_i}, respectively. If $\sum_{i=1}^{n}(P^{R_{cnd_i}} \cdot P^{R_{Q_i}}) < \rho$, the candidate object can be pruned.

Pruning rule 5 computes an upper bound of the RNN probability of the candidate object by assuming that all instances in R_{cnd_i} are RNNs of all instances in Q_i. The candidate object can be safely pruned if this upper bound is still less than the threshold.

Example 3: Fig. 12 shows MBRs of the query object R_Q and a candidate object R_{cnd} along with their instances (q_1 to q_5 and u_1 to u_4). Assume that all instances within an object have equal appearance probabilities (e.g; $p_{q_i} = 0.2$ for every q_i and $p_{u_i} = 0.25$ for every u_i). Suppose that no part of R_{cnd} can be pruned using R_Q and any filtering object R_{fil} (for better illustration, filtering object is not shown). We prune R_{cnd} using the rectangle R_{Q_1} that is contained by R_Q. This trims R_{cnd} and the remnant rectangle R_1 is obtained. Similarly, R_2 is the remnant rectangle when pruning rules are applied for R_{Q_2}. Note that only the instances in R_1 (u_1 and u_2) can be the RNN of instances in R_{Q_1} (q_3, q_4 and q_5). Similarly, no instance can be the RNNs of any instance in R_{Q_2} because R_2 is empty. So the maximum RNN probability of R_{cnd} is $(0.6 \times 0.5) + (0.4 \times 0) = 0.3$. If the probability threshold ρ is greater than 0.3, we can prune R_{cnd}. Otherwise, we can continue to trim R_{cnd} by using the smaller rectangles contained in R_{Q_1}. ■

5. We only require instances of Q to be disjoint. The pruning rule can be applied even when the minimum bounding rectangles R_{Q_i} overlap each other as shown in Fig. 12.

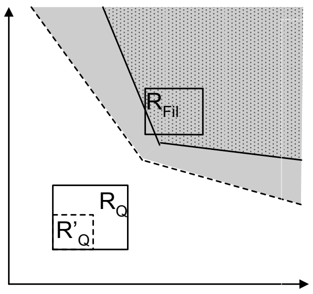


Fig. 11: Regions pruned by R_Q and its subset R'_Q

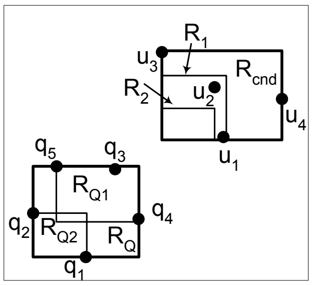


Fig. 12: Probabilistic Pruning

In our implementation, we build an R-tree on query object and the pruning rule is applied iteratively using MBRs of children. For more details, please see Algorithm 5.

Although the smaller rectangles R'_{fil} contained in R_{fil} can also be used, we do not use them because unlike query object there may be many filtering objects. Hence, using the smaller rectangles for each of the filtering objects would make this pruning rule very expensive in practice (more expensive than the efficient verification presented in Section 4.3).

3.5 Integrating the pruning rules

Algorithm 2 is the implementation of Pruning rules 1–4. Specifically, we apply pruning rules in increasing order of their computational costs (i.e., from Pruning rule 4 to 1). While simple pruning rules are not as restricting as more expensive ones, they can quickly discard many non-promising candidate objects and save the overall computational time.

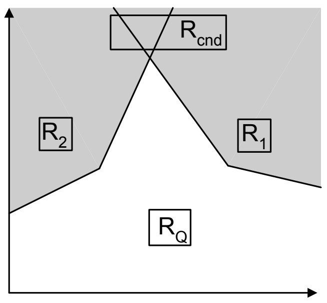


Fig. 13: R_{cnd} can be pruned by R_1 and R_2

Algorithm 2 : Prune(Q, S_{fil}, R_{cnd})

Input: R_Q: an MBR containing instances of Q ; S_{fil}: a set of MBRs to be used for trimming R_{cnd}: the candidate MBR to be trimmed

Description:

```
1: for each $R_{fil}$ in $S_{fil}$ do
2:    if $maxdist(R_{cnd}, R_{fil}) < mindist(R_Q, R_{cnd})$ then    // Pruning rule 4
3:       return $\phi$
4:    if $mindist(R_{cnd}, R_{fil}) > maxdist(R_Q, R_{cnd})$ then
5:       $S_{fil} = S_{fil} - R_{fil}$  // $R_{fil}$ cannot prune $R_{cnd}$
   $Rem = R_{cnd}$
6: for each $R_{fil}$ in $S_{fil}$ do
7:    if $R_{fil}$ is fully dominated by $R_Q$ in a partition $p$ then  // Pruning rule 3
8:       if some part of $Rem$ lies in the partition $p$ then
9:          $Rem =$ the part of $Rem$ not dominated by $F_p$
10:         if ($Rem = \phi$) then return $\phi$
11: for each $R_{fil}$ in $S_{fil}$ do
12:    $Rem =$ hspace_pruning($R_Q, R_{fil}, Rem$)  // Pruning Rules 1 and 2
13:    if ($Rem = \phi$) then return $\phi$
14: return $Rem$
```

It is important to use *all* the filtering objects to filter a candidate objects. Consider the example in Fig. 13. R_{cnd} cannot be pruned by either R_1 or R_2, but will be pruned by considering both of them.

Two subtle optimizations in the algorithm are:

- If $mindist(R_{cnd}, R_{fil}) > maxdist(R_Q, R_{cnd})$ for a given MBR R_{fil}, then R_{fil} cannot prune any part of R_{cnd}. Hence such R_{fil} is not considered for dominance and half-space pruning (lines 4-5). However, R_{fil} may still prune some other candidate objects, so we remove such R_{fil} only from a *local* set of filtering object, S_{fil}. This optimization reduces the cost of dominance and half-space pruning.
- If the frontier point F_{p_1} of a filtering object R_{fil_1} is dominated by the frontier point F_{p_2} of another filtering object R_{fil_2}, then F_{p_1} can be removed from S_{fil} because the area pruned by F_{p_1} can also be pruned by F_{p_2}. However, note that a frontier point cannot be used to prune its own rectangle. Therefore, before deleting F_{p_1}, we use it to prune rectangle belonging to F_{p_2}. This optimization reduces the cost of dominance pruning.

4 PROPOSED SOLUTION

In this section, we present our algorithm to find the probabilistic RNNs of an uncertain query object Q. The data is stored in system as follows: for each uncertain object, an R-tree is created and stored on disk that contains the instances of the uncertain object. Each node of the R-tree contains the aggregate appearance probability of the instances in its subtree. We refer these R-trees as *local* R-trees of the objects. Another R-tree is created that stores the MBRs of all uncertain objects. This R-tree is called *global R-tree*.

Algorithm 3 outlines our approach. Our algorithm consists of three phases namely Shortlisting, Refinement and Verification. In the following sub-sections, we present the details of each of these three phases.

Algorithm 3 : Answering Probabilistic RNN

Input: Q: uncertain query object; ρ: probability threshold;
Output: all objects that have higher than ρ probability to be RNN of Q
Description:

```
1: Shortlisting: Shortlist candidate and filtering objects (Algorithm 4)
2: Refinement: Trim candidate objects using disjoint subsets of $Q$ and apply pruning rule 5 (Algorithm 5)
3: Verification: Compute the exact probabilities of each candidate and report results
```

4.1 Shortlisting

In this phase (Algorithm 4), the global R-tree is traversed to shortlist the objects that may possibly be the RNN of Q. The MBR R_{cnd} of each shortlisted candidate object is stored in a set of candidate objects called S_{cnd}. Initially, root entry of the R-tree is inserted in a min-heap H. Each entry e is inserted in the heap with key $maxdist(e, R_Q)$ because a hyper-rectangle that has smaller maximum distance to R_Q is likely to prune a larger area and has higher chances to become the result.

Algorithm 4 : Shortlisting

```
1: $S_{fil} = \emptyset$, $S_{cnd} = \emptyset$
2: Initialize a min-heap $H$ with root entry of Global R-Tree
3: while H is not empty do
4:    de-heap an entry $e$
5:    if ($Rem$ = prune($R_Q, S_{fil}, e$)) $\neq \phi$ then
6:       if $e$ is a data object then
7:          $S_{cnd} = S_{cnd} \cup \{e\}$
8:       else if $e$ is a leaf or intermediate node then
9:          $S_{fil} = S_{fil} - \{e\}$
10:         for each data entry or child $c$ in $e$ do
11:            insert $c$ into $H$ with key $maxdist(c, R_Q)$
12:            $S_{fil} = S_{fil} \cup \{c\}$
```

We try to prune every de-heaped entry e (line 5) by using the pruning rules presented in the previous section. If e is a data object and cannot be pruned, we insert it into S_{cnd}. Otherwise, if e is an intermediate or leaf node, we insert its children c into heap H with key $maxdist(c, R_Q)$. Note that an entry e can be removed from S_{fil} (line 9) if at least one of its children is inserted in S_{fil} because the area pruned by an entry e is always contained by the area pruned by its child (Lemma 5 in our technical report [5]).

4.2 Refinement

In this phase (Algorithm 5), we refine the set of candidate objects by using pruning rule 5. More specifically, we descend into the R-tree of Q and trim each candidate object R_{cnd} against the children of Q and apply pruning rule 5.

Let P^R be the aggregate probability of instances in any hyper-rectangle R. At this stage $P^{R_{cnd}}$ of a candidate object may be less than one because R_{cnd} might have been trimmed during shortlisting phase. We can prune R_{cnd} if upper bound RNN probability of a candidate object $MaxProb = P^{R_{cnd}}$ is less than ρ (line 3).

We use a max-heap that stores entries in form (e, R, key) where e and R are hyper-rectangles containing instances of Q and R_{cnd}, respectively. key is the maximum probability of instances in R to be the RNNs of instances in e (i.e; $key = P^e \cdot P^{R_{cnd}}$). We initialize the heap by inserting (Q, R_{cnd},

Algorithm 5 : Refinement

Description:

```
1: for each $R_{cnd}$ in $S_{cnd}$ do
2:   if $(MaxProb = P^{R_{cnd}}) < \rho$ then
3:     $S_{cnd} = S_{cnd} - R_{cnd}$; continue;
4:   Initialize a max-heap H containing entries in form $(e, R, key)$
5:   insert $(Q, R_{cnd}, MaxProb)$ into $H$
6:   while H is not empty do
7:     de-heap an entry $(e, R, p)$
8:     $Rem$ = Prune $(e, S_{fil}, R)$
9:     $MaxProb = MaxProb - p + (P^e \cdot P^{Rem})$
10:    if $MaxProb < \rho$ then
11:      $S_{cnd} = S_{cnd} - R_{cnd}$; break;
12:    if ($P^{Rem} > 0$) AND ($e$ is an intermediate node or leaf) then
13:      for each child $c$ of $e$ do
14:        insert $(c, Rem, (P^c \cdot P^{Rem}))$ into H
```

$MaxProb$) (line 5). For each de-heaped entry (e, R, p), we trim the hyper-rectangle R against e by S_{fil} and store the trimmed rectangle in Rem (line 8). The upper bound RNN probability $MaxProb$ is updated to $MaxProb - p + (P^e \cdot P^{Rem})$. Recall that $p = P^e \cdot P^R$ was inserted with this entry assuming that all instances in R are RNNs of all instances in e. After we trim R using e (line 8), we know that only the instances in Rem can be RNNs of e. That is the reason we subtract p from $MaxProb$ and add $(P^e \cdot P^{Rem})$.

At any stage, if the $MaxProb < \rho$ the candidate object can be pruned. Otherwise, an entry $(c, Rem, (P^c.P^{Rem}))$ is inserted into the heap, for each child c of e. Note that if the trimmed hyper-rectangle does not contain any instance then P^{Rem} is zero and we do not need to insert children of e in the heap for such Rem.

Recall that every node in local R-tree stores the aggregate appearance probability of all instances in its sub-tree which makes computation of aggregate probability cheaper.

4.3 Verification

The actual probability of a candidate object R_{cnd} to be the RNN of Q is the sum of probabilities of every instance $u_i \in R_{cnd}$ to be the RNN of every instance q of Q . To compute the probability of an instance u_i to be RNN of q, we have to find, for each uncertain object U, the accumulative appearance probability of its instances that have smaller distance to u_i than $dist(q, u_i)$ (Equation (2)). A straight forward approach is to issue a range query for every $u_i \in R_{cnd}$ centred at u_i with range set as $dist(q, u_i)$ and then compute the accumulative appearance probability of instances of each object that are returned. However, this approach requires $\mid Q \mid \times \mid R_{cnd} \mid$ number of range queries where $\mid Q \mid$ and $\mid R_{cnd} \mid$ are number of instances in Q and R_{cnd}, respectively. Below, we present an efficient approach that issues only one global range query to compute the exact RNN probability of a candidate object.

4.3.1 Finding range of the global range query

Let R_{fil} be an MBR containing instances of a filtering object. An instance u_i has zero probability to be RNN of an instance q if $dist(u_i, q) > maxdist(u_i, R_{fil})$. So the range of a range query for u_i centred at u_i is minimum of $maxdist(u_i, R_Q)$ and $maxdist(u_i, R_{fil})$ for every R_{fil} in S_{fil}.

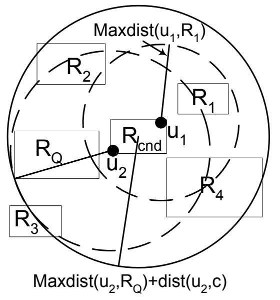


Fig. 14: Finding the range of the global query

Consider the example of Fig. 14 where the range of queries centred at u_1 and u_2 are $maxdist(u_1, R_1)$ and $maxdist(u_2, R_Q)$, respectively (circles with broken lines).

We want to reduce multiple range queries to a single range query centred at the centre of R_{cnd} with a *global range* r such that all instances required to compute RNN probability of every candidate instance $u_i \in R_{cnd}$ are returned. Let r_i be the range of the range query of u_i computed as described above. The global range r is $\max(r_i + dist(u_i, c))$ for every $u_i \in R_{cnd}$ where c is the centre of R_{cnd}. In the example of Fig. 14, the global range is $r = maxdist(u_2, R_Q) + dist(u_2, c)$ as shown in the figure (solid circle). Note that this range ensures that all the instances required to compute RNN probability of both u_1 and u_2 lie within this range.

4.3.2 Computing the exact RNN probability of R_{cnd}

We issue a range query on global R-tree with range r as computed above. For each returned object U_i, we issue a range query on the local R-tree of U_i to get the instances that lie within the range and then create a list L_i containing all these instances. We sort the entries in each list L_i in ascending order of their distances from u_{cnd}.

The list L_Q for the instances of query object Q is shown in Fig. 15. Each entry e contains two values (d, p) such that d is distance of e from u_{cnd} and p is the appearance probability of the instance e. The lists for other objects are slightly different in that each entry e contains two values (d, P) where P is the *accumulative* appearance probability of all the instances that appear in the list before e. In other words, given an entry (d, P), the total appearance probability of all instances (in this list) that have smaller distance than d is P.

Given these lists, we can quickly find the accumulative appearance probability of all instances of any uncertain object that lie closer to u_{cnd} than a query instance q_i. The example below illustrates the computation of exact probability of a candidate instance u_{cnd}.

Example 4: Fig. 15 shows the lists of query object Q and three uncertain objects A, B and C. The lists are sorted on their distances from the candidate instance u_{cnd}. We start the computation from the first entry q_2 in Q and compute $RNN_{q_2}(u_{cnd})$. The distance d_{q_2} is 0.3. We do a binary search on A, B and C to find an entry in each list with largest d smaller than d_{q_2}. Such entries are $a_3(0.1, 0.3)$ and $b_4(0.2, 0.4)$ in lists A and B, respectively. No instance is found in C. Hence, the sum of appearance probabilities of

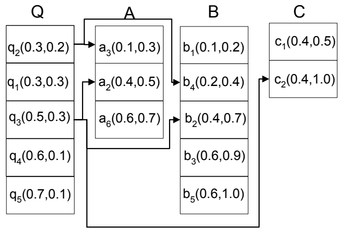


Fig. 15: lists sorted on distance from a candidate instance u_{cnd}

instances of B that have distance from u_{cnd} smaller than d_{q_2} is 0.4, similarly for A it is 0.3. Given both q_2 and u_{cnd} appear in a world, the probability of u_{cnd} to be RNN of q_2 is obtained from Equation (2) as $(1 - 0.4)(1 - 0.3) = 0.42$. The probability of u_{cnd} to be RNN of q_2 in any possible world is $0.42(p_{q_2} \times p_{u_{cnd}})$.

Similarly the next entry in Q is processed and $RNN_{q_1}(u_{cnd})$ is computed which is again 0.42 because its distance from u_{cnd} is the same. $RNN_{q_3}(u_{cnd})$ is zero because the binary search on C gives an entry (d, P) where $P = 1$ (all instances of C have smaller distance to u_{cnd} then d_{q_3}). Note that, we do not need to compute the RNN probabilities of u_{cnd} against remaining instances q_4 and q_5 because their distances from u_{cnd} are larger than d_{q_3} and $RNN_{q_3}(u_{cnd}) = 0$. Also note that the area to be searched in any list L_i by binary search becomes smaller for the processing of next query instance. ∎

The above example illustrates the probability computation of an instance u_{cnd} to be the RNN of all instances in Q. We repeat this for every instance $u_{cnd} \in R_{cnd}$ to compute the RNN probability of the candidate object. Next, we present some optimizations that improve the efficiency of verification phase.

Optimizations

Our proposed optimizations bound the minimum and maximum RNN probabilities and verify the objects that have the minimum probability greater than or equal to the threshold. Similarly, the objects that have the maximum probability less than the threshold are deleted. Below, we present the details of the proposed optimizations.

a) Bounding RNN probabilities using R_Q:

Recall that, for each candidate object R_{cnd}, a global range query is issued and for each object U_i within the range a list L_i is created containing the instances of U_i lying within the range. Just before we sort these lists, we can approximate the maximum and minimum RNN probability of the candidate object based on the following observations.

Let c be the centre and d be the diagonal length of R_{cnd} and a_i be some instance in list A. Every $u_{cnd} \in R_{cnd}$ is always closer to a_i than every $q_i \in Q$ if $mindist(R_{cnd}, R_Q) > dist(a_i, c) + d/2$. Similarly, every u_{cnd} would always be further from a_i than every $q_i \in Q$ if $maxdist(R_{cnd}, R_Q) < dist(a_i, c) - d/2$. Consider the example of Fig. 16, every point in R_{cnd} is always closer to a_1 than any point in R_Q. Similarly, every point in R_{cnd} is always further from a_2 than it is from any point in R_Q.

Based on the above observations, for every object, we can accumulate the appearance probabilities of all the instances u such that every u_{cnd} is always closer to (or further from) u than every q_i. More specifically, we traverse each list L_i and accumulate the appearance probabilities of every instance u_i for which $mindist(R_{cnd}, R_Q) > dist(u_i, c) + d/2$ and store the accumulated probabilities in P_i^{near}. Similarly, the accumulated appearance probabilities of every instance u_j for which $maxdist(R_{cnd}, R_Q) < dist(u_j, c) - d/2$ is stored in P_i^{far}. Then the maximum RNN probability of any instance u_{cnd} is $p_{cnd}^{max} = \prod_{\forall L_i}(1 - P_i^{near})$. The minimum probability of any instance u_{cnd} to be RNN of Q is $p_{cnd}^{min} = \prod_{\forall L_i}(P_i^{far})$ because P_i^{far} is the total probability of instances that are definitely farther. So we assume that all other instances are closer to u_{cnd} than q_i and this gives us the minimum RNN probability.

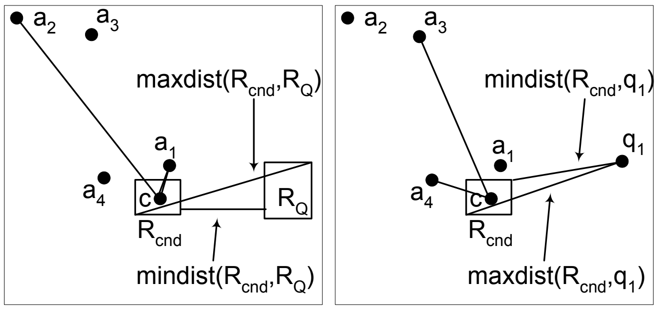


Fig. 16: Bounding lower and upper bound RNN probabilities

Let $P^{R_{cnd}}$ be the aggregate appearance probability of all the instances in R_{cnd} then R_{cnd} can be pruned if $P^{R_{cnd}} \cdot p_{cnd}^{max} < \rho$. Similarly, the object can be reported as answer if $P^{R_{cnd}} \cdot p_{cnd}^{min} \geq \rho$.

b) Bounding RNN probabilities using instances of Q:

If an object R_{cnd} cannot be pruned or verified as result at this stage, we try to make a better estimate of p_{cnd}^{min} and p_{cnd}^{max} by using instances within Q. Note that every $u_{cnd} \in R_{cnd}$ is always closer to a_i than a query instance q_i if $mindist(R_{cnd}, q_i) > dist(a_i, c) + d/2$. Similarly, every u_{cnd} would always be further from a_i than q_i if $maxdist(R_{cnd}, q_i) < dist(a_i, c) - d/2$. Consider the example of Fig. 16 where every point in R_{cnd} is closer to both a_1 and a_4 than q_1. Similarly, every point in R_{cnd} is further from both a_2 and a_3 than it is from q_1.

To update p_{cnd}^{max}, we first sort every list in ascending order of $dist(c, u)$ where $dist(c, u)$ is already known (returned by global range query). Then, the list L_Q is sorted in ascending order of the $mindist(R_{cnd}, q_i)$. Then for each q_i in ascending order, we conduct a binary search on every list L_i and find the entry $e(d, P)$ with greatest d in the list that is less than $mindist(R_{cnd}, q_i) - d/2$. The probability P of this entry is accumulated appearance probability P_i^{near} of all the instances a_i such that every u_{cnd} is always closer to a_i than q_i. Then the maximum probability of any instance $u_{cnd} \in R_{cnd}$ to be the RNN of q_i is $p_{i_{cnd}}^{max} = \prod_{\forall L_i}(1 - P_i^{near})$. We do such binary searches for every q_i in the list and $p_{cnd}^{max} = \sum_{\forall q_i \in Q} p_{i_{cnd}}^{max}$.

The update of p_{cnd}^{min} is similar except that the list L_Q is sorted in ascending order of $maxdist(R_{cnd}, q_i)$ and the binary search is conducted to find the entry $e(d,P)$ with the greatest d that is smaller than $maxdist(R_{cnd},q_i) + d/2$. The total appearance probabilities of all instances in L_i that are always farther from every u_{cnd} than q_i is $P_i^{far} = (1-P)$. Finally, $p_{i_{cnd}}^{min} = \prod_{\forall L_i}(P_i^{far})$ and $p_{cnd}^{min} = \sum_{\forall q_i \in Q} p_{i_{cnd}}^{min}$.

After updating p_{cnd}^{max} and p_{cnd}^{min}, we delete the candidate objects for which $P^{R_{cnd}} \cdot p_{cnd}^{max} < \rho$. Similarly, a candidate object is reported as answer if $P^{R_{cnd}} \cdot p_{cnd}^{min} \geq \rho$.

c) Early stopping:

If an object R_{cnd} is not pruned by the above mentioned estimation of maximum and minimum RNN probabilities then we have to compute exact RNN probabilities (as described in Section 4.3.2) of the instances in it. By using the maximum and minimum RNN probabilities, it is possible to verify or invalidate an object without computing the exact RNN probabilities of all the instances. We achieve this as follows; We sort all the instances in R_{cnd} in descending order of their appearance probabilities. Assume that we have computed the exact RNN probability $RNN_Q(u)$ of first i instances. Let P be the aggregate appearance probabilities of these first i instances and P_{RNN} be the sum of their $RNN_Q(u)$. At any stage, an object can be verified as answer if $P_{RNN} + (1-P).p_{cnd}^{min} \geq \rho$. Similarly, an object can be pruned if $P_{RNN} + (1-P).p_{cnd}^{max} < \rho$.

Note that $(1-P).p_{cnd}^{min}$ is the minimum probability for the rest of the instances to be the RNN of Q. Similarly, $(1-P).p_{cnd}^{max}$ is the maximum probability for the remaining instances to be the RNN.

5 EXPERIMENT RESULTS

In this section we evaluate the performance of our proposed approach. All the experiments were conducted on Intel Xeon 2.4 GHz dual CPU with 4 GBytes memory. The node size of each local R-tree is $1K$ and that of global R-tree is $2K$. *We measured both the I/O and CPU time and I/O cost is around 1-5$\%$ of the total cost for all experiments.* Hence, for clarity of experiment figures, we display the average total cost per query. We used both synthetic and real datasets.

TABLE 2: System Parameters

Parameter	Range
Probability threshold (ρ)	0.1, 0.3, **0.5**, 0.7, 0.9
Number of objects ($\times$1000)	2, 4, **6**, 8, 10
Maximum number of instances in an object	200, 400, **600**, 800, 1000
Maximum width of hyper-rectangle	1%, **2%**, 3%, 4%
Distribution of object centres	**Uniform**, Normal
Distribution of instances	**Uniform**, Normal
Appearance probability of instances	**Uniform**, Normal

Table 2 shows the specifications of the synthetic datasets we used in our experiments and the defaults values are shown in bold. First the centres of the uncertain objects were created (uniform or normal distribution) and then the instances for each object (uniform or normal distribution) were created within their respective hyper-rectangles. The width of the hyper-rectangle in each dimension was set from 0 to $w\%$ (following uniform distribution) of the whole space and we conducted experiments for w changed from 1 to 4. The appearance probabilities of instances were generated following either uniform or normal distribution. Our default synthetic dataset contains approximately 1.8 Million instances ($\frac{6000\times600}{2}$). Similar to [19], the query object follows same distribution as the underlying dataset.

The real dataset[6] consists of 28483 zip codes obtained from 40 states of United States. Each zip code represents an object and the *address blocks* within each zip code are the instances. The data source provides address ranges instead of individual addresses and we use the term *address block* for a range of addresses along a road segment. The address block is an instance in our dataset that lies at the middle of the road segment with the appearance probability calculated as follows; Let n be the number of total addresses in a zip code and m be the number of addresses in the current address block then the appearance probability of the current address block is m/n. The real dataset consists of 11.24 Million instances and the maximum number of instances (address blocks) in an object (Sanford, North Carolina) were 5918.

5.1 Comparison with other possible solutions

We devise a naïve algorithm and a sampling based approximate algorithm to better understand the performance of our algorithm. More specifically, in the naïve algorithm, we first shortlist the objects using our pruning rule 4 (e.g; any object R_{cnd} can be pruned if $mindist(R_{cnd}, R_Q) > maxdist(R_{cnd}, R_{fil})$). Then, we verify the remaining objects as follows. For each pair (u_i, q_i), we issue a range query centred at u_i with range $dist(u_i, q_i)$ and compute the RNN probability of the instance u_i against the query instance q_i using the Equation (2). Finally, the Equation (1) is used to compute the RNN probability of the object.

In sampling based approach, we create a few sample possible worlds before starting the computation. More specifically, a possible world is created by randomly selecting one instance from each uncertain object. For each possible world, we create an R-tree (node size $2K$) that stores the instances of the possible worlds. This reduces the problem of finding probabilistic RNNs to conventional RNNs. For each possible world, we compute the RNNs using TPL [20] that is the best-known RNN algorithm for multidimensional data. Let n be the number of possible worlds evaluated and m be the number of possible worlds in which an object R_{cnd} is returned as RNN, then R_{cnd} is reported as answer if $m/n \geq \rho$. The costs shown do not consider the time taken in creating the possible worlds. Note that this algorithm provides only approximate results. For real dataset, the accuracy varies from 60% to 75%.

Naïve algorithm appeared to be too slow (average query time from 7 minutes to 2 hours) so we show its computation time only when comparing our verification phase in Fig. 18.

Fig. 17 compares our approach with the sampling based approximate approach (for 100 and 200 possible worlds) on synthetic dataset. In two dimensional space, our algorithm is comparable with the sampling algorithm that returns approximate answer. On the other hand, the Fig. 17 shows that our algorithm is more efficient for higher dimensions and scales

6. http://www.census.gov/geo/www/tiger/

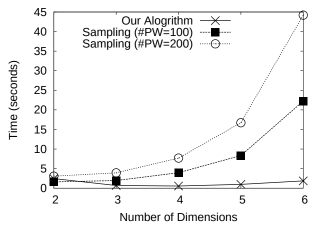


Fig. 17: Overall cost

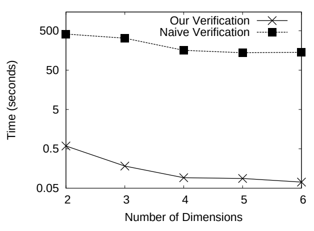


Fig. 18: Verification cost

better. The cost for our algorithm first decreases as the number of dimensions increase and then it starts increasing. The reason is that for low dimensional space, the data is more dense and the verification phase cost dominates the pruning phase cost. On the other hand, for high-dimensional space, the data is sparse and while the verification is cheaper the pruning phase is expensive (e.g; greater number of bisectors required to prune the space).

In Fig. 18, we compare the verification cost of our algorithm with the verification cost of naïve algorithm. The costs shown are verification costs per candidate object. Our proposed verification is three orders of magnitude faster than the naïve verification.

5.2 Performance on real dataset and effect of data distribution

Fig. 19 compares the performance of our algorithm against the sampling based approximate algorithm on real dataset for probability threshold changed from 0.1 to 0.9. For sampling based algorithm, the costs are shown for the evaluation of 100 and 200 possible worlds. Our algorithm performs better than the approximate sampling based algorithm for larger threshold.

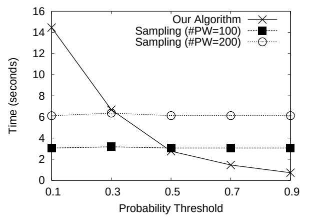


Fig. 19: Comparison on Real Dataset

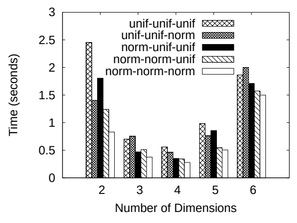


Fig. 20: Effect of data distribution

Note that although the accuracy may vary, the cost of sampling algorithm does not change with the change in threshold, underlying data distribution (as noted in [20]), width of hyper-rectangle or number of instances in each object. Moreover, the cost of sampling algorithm increases linearly with the number of possible worlds evaluated. For this reason, now we focus on the performance evaluation of only our proposed algorithm.

Fig. 20 shows the performance of our algorithm for different data distributions. The legend shows data distributions in form dist1_dist2_dist3 where dist1 is the distribution of the object centres, dist2 is the distribution of instances within the objects and dist3 is the distribution of appearance probability. For example, norm_norm_unif shows the result for the data such that the centres of objects and instances are normally distributed with appearance probability following uniform distribution. The performance of our algorithm on non-uniform data is better than the uniform data as can be observed from Fig. 20. This is mainly due to two reasons. Firstly, we observe that the number of candidates in S_{cnd} is smaller after the pruning phase if the data is non-uniform. Secondly, if the probability distribution is not uniform the verification phase is faster because we sort the instances in descending order of their appearance probabilities and this lets us validate or invalidate an object earlier.

5.3 Effect of data size

In Fig. 21, we increase the maximum number of instances in each object from 200 to 1000. The performance degrades as the number of instances increase. Although the increase in number of instances does not have significant effect on pruning phase, the verification phase becomes more expensive if each object has greater number of instances. Also observe that the cost does not change significantly for higher dimensions because in high dimensional space, the pruning phase cost is dominant which is not affected significantly by the number of instances

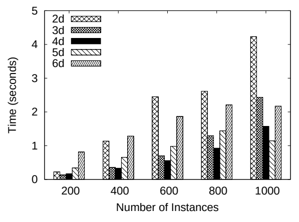


Fig. 21: Effect of number of instances in each object

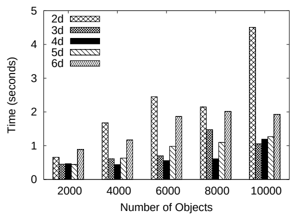


Fig. 22: Effect of number of objects in the dataset

Fig. 22 evaluates the performance of our algorithm with increasing number of objects in the dataset. The computation cost increases with increase in number of objects mainly due to the increased verification cost because larger number of objects (and in effect instances) are returned by the global range query.

5.4 Effect of probability threshold and width of hyper-rectangle

Fig. 23 shows the effect of probability threshold. The algorithm performs better as the probability threshold ρ increases because fewer number of candidate objects pass the pruning phase and require the verification. The effect is more significant in lower dimensions because for low dimensions the verification cost dominates the overall cost.

In Fig. 24, we change width of each hyper-rectangle and study the performance of our algorithm. The performance degrades in low-dimensional space due to larger overlap of objects with each other and the query object. The effect in higher dimensions is not as significant as in low-dimensional space.

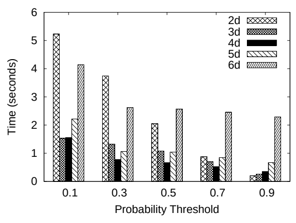


Fig. 23: Effect of probability Threshold

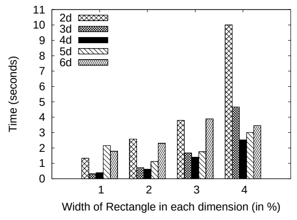


Fig. 24: Effect of width of hyper-rectangles

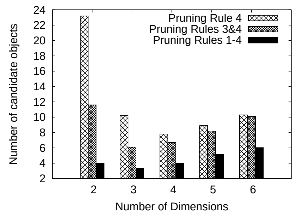


Fig. 27: Effectiveness of pruning rules

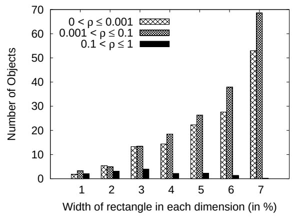


Fig. 28: Effect of width of hyper-rectangles

5.5 Evaluation of different phases

In this section, we study the effect of our pruning phases. More specifically, we compare the number of candidates after first phase (shortlisting), second phase (refinement), optimization (of the verification phase) and the number of objects in final result. Fig. 25 shows the number of candidates after each phase. The number of candidates after *shortlisting* is from 10-20 and the *refinement* phase reduces the number to less than its half. The optimization presented in the verification phase prunes more objects in high-dimensional space because in low-dimensional space due to larger volume of MBRs, most of the MBRs of remaining candidates overlap with the query object. Hence the optimizations are more useful for higher dimensions.

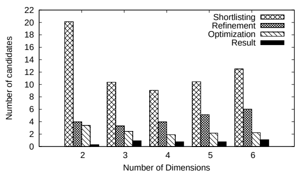


Fig. 25: Number of objects in S_{cnd} after each phase

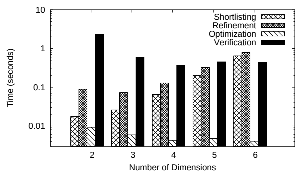


Fig. 26: Computational time taken by each phase

Fig. 26 shows the time taken by each of the pruning phase. Our proposed optimization takes very small amount of time and is quite useful especially for high-dimensional data. Verification phase is the dominant cost for low-dimensional queries and the pruning phases (shortlisting and refinement) dominate the overall cost for high-dimensional queries. Note that logscale is used for y-axis.

5.6 Effectiveness of pruning rules

Pruning rule 5 is used in phase 2 (refinement) of our algorithm and uses the other pruning rules to estimate the maximum probability. Its effectiveness can be observed in Fig. 25 by comparing the number of objects after *shortlisting* and *refinement* phases.

Fig. 27 shows the effectiveness of other pruning rules. We observed that the dominance pruning rule prunes fewer objects than the simple distance based pruning rule 4. However, the dominance pruning can prune some objects that cannot be pruned by the simple pruning rule because the dominance pruning rule can trim part of the candidate objects.

Fig. 27 shows the number of candidates after *refinement* phase of our algorithm when a combination of pruning rules is used. More specifically, we compare the number of objects in S_{cnd} when only the pruning rule 4 is used, the dominance pruning is used along with pruning rule 4, and when all pruning rules from 1 to 4 are used. Since pruning rule 5 uses the other underlying pruning rules, it is enabled for all above mentioned settings. The half-space pruning significantly reduces the number of candidate objects and the effectiveness of dominance pruning is more significant for the low-dimensional data.

5.7 Effect of hyper-rectangle width on the size of result

We note that if the hyper-rectangles of objects largely overlap each other, the probabilistic reverse nearest neighbor queries are not very meaningful. In other words, there would be no objects satisfying some reasonable probability threshold (a value that can be considered significant). Fig. 28 shows the number of objects that satisfy different probability thresholds. The width of hyper-rectangle in each dimension is changed from 1% to 7% and the results are shown for two dimensional space. It can be observed that with large overlap in rectangles, more and more objects satisfy very small probability threshold constraint. On the other hand, there are very few or no object at all that have greater than 0.1 probability to be the RNN.

6 Related Work

Recently, a lot of work has been dedicated to uncertain databases (see The TRIO system [22], The ORION project [7] and the references therein). Query processing on uncertain databases has gained significant attention in last few years especially in spatio-temporal databases.

In [8], the authors develop index structures to querying uncertain interval effectively. They are the first to study probabilistic range queries. In [19], the authors propose access methods designed to optimize both the I/O and CPU cost of range queries on multi-dimensional data with arbitrary probability density functions. The concept of probabilistic similarity joins on uncertain objects is first introduced in [13] which assigns a probability value to each object pair indicating the likelihood that it belongs to the result set. *Ranking* and *thresholding* probabilistic spatial queries are studied in [9]. A thresholding probabilistic query is to retrieve the objects qualifying the spatial predicates with probability greater than a given threshold. Similarly, a ranking probabilistic query retrieves the objects with the highest probabilities to qualify the spatial predicates. A probabilistic skyline model is proposed in [17] alongwith two effective algorithms to answer probabilistic skyline queries. While nearest neighbor queries on uncertain objects are studied in [4], [6], [14], to the best of our knowledge, there does not exist any previous work on reverse nearest neighbor queries on uncertain data.

Now, we overview the previous work related to reverse nearest neighbor queries where the data is not uncertain. Korn *et. al* [12] are first to introduce the reverse nearest neighbor queries. They provide a solution based on the pre-computation of the nearest neighbor of each data point. More efficient solutions based on pre-computation are proposed in [26] and [15]. Stanoi *et. al* proposed a method that does not require any pre-computation. They observe that in $2d$-space, the space around query can be partitioned into six equal regions and only the nearest neighbor of query in each region can be the RNN of the query. However, the number of regions to be searched for candidate objects increases exponentially with the dimensionality. Singh *et al.* [18] propose a solution that performs better in high-dimensional space. They first find K (system parameter) nearest neighbors of the query object and then check whether the retrieved objects are the RNNs of query object or not. Tao *et al.* [20] utilize the idea of perpendicular bisector to reduce the search space. They progressively find nearest neighbors of query and for each nearest neighbor they draw a perpendicular bisector that divides the space in two partitions. Only the objects that lie in the partition containing query object can be the reverse nearest neighbors. Recently, Wu *et. al* [24] propose an algorithm for RkNN queries in $2d$-space. Instead of using bisectors to prune the objects, they use a convex polygon obtained from the intersection of bisectors. Any object that lies outside the polygon can be pruned.

Continuous monitoring of RNN queries is studied in [3], [11], [23], [25]. Reverse nearest neighbors in metric spaces ([1], [2], [21]), large graphs [28] and ad hoc subspaces [27] has also been explored. Problem of reverse nearest neighbor aggregates over data streams is studied in [30]. Other variants like ranked reverse nearest neighbor queries and reverse furthest nearest neighbor queries are studied in [31] and [29], respectively.

7 Conclusion

In this paper, we studied the problem of reverse nearest neighbor queries on uncertain data and proposed novel pruning rules that effectively prune the objects that cannot be the RNNs of query. We proposed an efficient algorithm and presented several optimizations that significantly reduce the overall computation time. Using real dataset and synthetic dataset, we illustrated the efficiency of our proposed approach. Although we focused on discrete case, the pruning rules we presented can be applied when the uncertain objects are represented by probability density function. As future work, we will study the extension of our solution to probabilistic RkNN queries on uncertain data.

Acknowledgments

The work of Xuemin Lin is supported by Australian Research Council Discovery Grants (DP0987557, DP0881035 and DP0666428) and Google Research Award. Wei Wang's research is supported by ARC Discovery Grants DP0987273 and DP0881779. Jian Pei's research is supported in part by a NSERC Discovery grant and a NSERC Discovery Accelerator Supplement grant.

References

[1] E. Achtert, C. Bohm, P. Kroger, P. Kunath, A. Pryakhin, and M. Renz. Approximate reverse k-nearest neighbor queries in general metric spaces. In *CIKM*, pages 788–789, 2006.

[2] E. Achtert, C. Böhm, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In *SIGMOD Conference*, pages 515–526, 2006.

[3] R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis. Nearest neighbor and reverse nearest neighbor queries for moving objects. In *IDEAS*, pages 44–53, 2002.

[4] G. Beskales, M. Soliman, and I. F. Ilyas. Efficient search for the top-k probable nearest neighbors inuncertain databases. In *VLDB*, 2008.

[5] M. A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei. Probabilistic reverse nearest neighbor queries on uncertain data. In *UNSW Technical Report, 2008. Available at ftp://ftp.cse.unsw.edu.au/pub/doc/papers/UNSW/0816.pdf*.

[6] R. Cheng, J. Chen, M. F. Mokbel, and C.-Y. Chow. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *ICDE*, pages 973–982, 2008.

[7] R. Cheng, S. Prabhakar, and D. V. Kalashnikov. Querying imprecise data in moving object environments. In *ICDE*, pages 723–725, 2003.

[8] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *VLDB*, pages 876–887, 2004.

[9] X. Dai, M. L. Yiu, N. Mamoulis, Y. Tao, and M. Vaitis. Probabilistic spatial queries on existentially uncertain data. In *SSTD*, pages 400–417, 2005.

[10] J. Goldstein, R. Ramakrishnan, U. Shaft, and J.-B. Yu. Processing queries by linear constraints. In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 257–267, New York, NY, USA, 1997. ACM.

[11] J. M. Kang, M. F. Mokbel, S. Shekhar, T. Xia, and D. Zhang. Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors. In *ICDE*, pages 806–815, 2007.

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

[13] H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz. Probabilistic similarity join on uncertain data. In *DASFAA*, pages 295–309, 2006.

[14] H.-P. Kriegel, P. Kunath, and M. Renz. Probabilistic nearest-neighbor query on uncertain objects. In *DASFAA*, pages 337–348, 2007.

[15] K.-I. Lin, M. Nolen, and C. Yang. Applying bulk insertion techniques for dynamic reverse nearest neighbor problems. *ideas*, 00:290, 2003.

[16] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: Query processing for location services without compromising privacy. In *VLDB*, pages 763–774, 2006.

[17] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *VLDB*, pages 15–26, 2007.

[18] A. Singh, H. Ferhatosmanoglu, and A. S. Tosun. High dimensional reverse nearest neighbor queries. In *CIKM*, pages 91–98, 2003.

[19] 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.

[20] Y. Tao, D. Papadias, and X. Lian. Reverse knn search in arbitrary dimensionality. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 744–755. VLDB Endowment, 2004.

[21] Y. Tao, M. L. Yiu, and N. Mamoulis. Reverse nearest neighbor search in metric spaces. *IEEE Trans. Knowl. Data Eng.*, 18(9):1239–1252, 2006.

[22] J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR*, pages 262–276, 2005.

[23] W. Wu, F. Yang, C. Y. Chan, and K.-L. Tan. Continuous reverse k-nearest-neighbor monitoring. In *MDM*, pages 132–139, 2008.

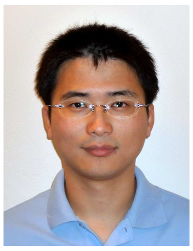
[24] W. Wu, F. Yang, C. Y. Chan, and K.-L. Tan. Finch: Evaluating reverse k-nearest-neighbor queries on location data. In *VLDB*, 2008.

[25] T. Xia and D. Zhang. Continuous reverse nearest neighbor monitoring. In *ICDE*, page 77, 2006.

[26] C. Yang and K.-I. Lin. An index structure for efficient reverse nearest neighbor queries. In *Proceedings of the 17th International Conference on Data Engineering*, pages 485–492, Washington, DC, USA, 2001. IEEE Computer Society.
[27] M. L. Yiu and N. Mamoulis. Reverse nearest neighbors search in ad hoc subspaces. *IEEE Trans. Knowl. Data Eng.*, 19(3):412–426, 2007.
[28] M. L. Yiu, D. Papadias, N. Mamoulis, and Y. Tao. Reverse nearest neighbors in large graphs. In *ICDE*, pages 186–187, 2005.
[29] B. Yao, F. Li, P. Kumar. Visible reverse k-nearest neighbor queries. In *ICDE*, 2009.
[30] F. Korn,S. Muthukrishnan, D. Srivastava. Reverse nearest neighbor aggregates over data streams. In *VLDB*, pages 814–825, 2002.
[31] K. C. K. Lee, B. Zheng, W. C. Lee. Ranked reverse nearest neighbor search. *IEEE Trans. Knowl. Data Eng.*, 20(7):894–910, 2008.

Muhammad Aamir Cheema is currently a PhD student in the School of Computer Science and Engineering, the University of New South Wales, Australia. He completed his Master of Engineering degree in Computer Science and Engineering from the University of New South Wales, Australia, in 2007. He received his B.Sc. degree in Electrical Engineering from University of Engineering and Technology, Lahore, in 2005. His current research interests include spatio-temporal databases, location-based services, mobile and pervasive computing and probabilistic databases. He served as Lecturer in James Cook University (Sydney campus) from 2006-2007. He also served as Associate Lecturer at the University of New South Wales, Australia, in 2007. (http://www.cse.unsw.edu.au/~macheema)

Xuemin Lin is a Professor in the School of Computer Science and Engineering, the University of New South Wales. He has been the head of database research group at UNSW since 2002. Before joining UNSW, Xuemin held various academic positions at the University of Queensland and the University of Western Australia. Dr. Lin got his PhD in Computer Science from the University of Queensland in 1992 and his BSc in Applied Math from Fudan University in 1984. During 1984-1988, he studied for PhD in Applied Math at Fudan University. He currently is an associate editor of ACM Transactions on Database Systems. His current research interests lie in data streams, approximate query processing, spatial data analysis, and graph visualization. (http://www.cse.unsw.edu.au/~lxue)



Wei Wang is currently a Senior Lecturer at the School of Computer Science and Engineering at University of New South Wales, Australia. He received his Ph.D. degree in Computer Science from Hong Kong University of Science and Technology in 2004. His research interests include integration of database and information retrieval techniques, similarity search, and query processing and optimization. (http://www.cse.unsw.edu.au/~weiw)

Wenjie Zhang is currently a PhD student in the School of Computer Science and Engineering, the University of New South Wales, Australia. She received her M.S. degree and B.S. degree both in computer science from Harbin Institute of Technology, China. Her research focuses on uncertain data management and spatio-temporal indexing techniques. She has published papers in conferences and journals including SIGMOD, ICDE and VLDBJ. She is also the recipient of Best Paper Award of National DataBase Conference of China 2005 and APWebWAIM 2009. (http://www.cse.unsw.edu.au/~zhangw)

Jian Pei is currently an Associate Professor and the director of Collaborative Research and Industry Relations at the School of Computing Science at Simon Fraser University. His research interests can be summarized as developing effective and efficient data analysis techniques for novel data intensive applications. His research has been supported extensively by government funding agencies and industry companies. He has published prolifically in refereed journals, conferences, and workshops. He has served regularly in the organization committees and the program committees of many international conferences and workshops. He is a senior member of ACM and IEEE. He is the recipient of the British Columbia Innovation Council 2005 Young Innovator Award, an NSERC 2008 Discovery Accelerator Supplements Award, an IBM Faculty Award (2006), and the KDD'08 Best Application Paper Award. (http://www.cs.sfu.ca/~jpei)