Probabilistic Reverse Nearest Neighbor Queries on Uncertain Data

Muhammad Aamir Cheema[1] Xuemin Lin[1] Wei Wang[1]
Wenjie Zhang[1] Jian Pei[2]

[1] University of New South Wales, Australia
{macheema,lxue,weiw,zhangw}@cse.unsw.edu.au

[2] School of Computer Science, Simon Fraser University, Canada
jpei@cs.sfu.ca

Technical Report
UNSW-CSE-TR-0816
July 2008

Abstract

Uncertain data is inherent in many important applications where the exact data values are not known. While many types of queries on uncertain data have been studied, reverse nearest neighbor query on uncertain data is still an open problem. In this paper, we formalize the problem of probabilistic reverse nearest neighbor query based on the possible worlds semantics. We propose an efficient method that processes such queries efficiently. The key technique innovation is several novel pruning methods that exploit various properties of the problem. Extensive experiment demonstrates that our algorithm is highly efficient and scalable.

1 Introduction

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.

Given the importance of these applications, query processing on uncertain data has gained much attention [1, 2]. Most recently, probabilistic nearest neighbor queries on uncertain data have been studied in [3]. However, to the best of our knowledge, there does not exist any prior work on probabilistic reverse nearest neighbor (RNN) queries on uncertain data. Probabilistic RNN queries have many applications. Consider the example of stock markets where 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.

Probabilistic RNN queries are also important for privacy preserving location-based services where the location of every user is obfuscated into a cloaked spatial region [4]. 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.

There exist various applications where the users might be interested in RNNs but only the probabilistic information of objects is available. For example, a probabilistic RNN query is issued if we want to find RNNs of users where we only know the zip codes of the users but not their exact addresses.

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 [5] 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 geometric properties and the probability threshold. (ii) a highly optimized verification method that is based on careful upper and lower bounding of the RNN probability of candidate objects. To better understand the performance of our proposed approach, we have conducted extensive experiments on the performance of our algorithm on both synthetic and real datasets. We show that our proposed algorithm is much more efficient than a baseline exact algorithm and performs better than a sampling-based approximate algorithm in most of the cases.

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.
- Experiment results on synthetic and real datasets show that our algorithm is much more efficient than a baseline algorithm and performs better than an 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. Terms are defined in Glossary (Section 8) and the proofs are presented in Appendix (Section 9).

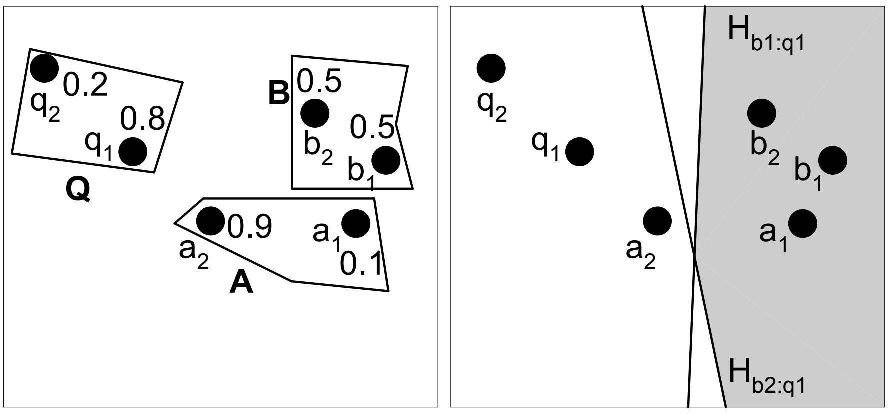


Figure 1.1: Example of a Probabilistic RNN query

2 Problem Definition and Preliminaries

2.1 Problem Definition

Given a set of *uncertain objects* $\mathcal{U} = \{U_1, ..., U_n\}$. Each uncertain object U_i consists of many *instances* $\{u_1, ..., u_m\}$. Each instance u_j is associated with a probability p_{u_j} called *appearance probability* with the constraint that $\sum_{j=1}^{m} p_{u_j} = 1$. 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;

$$RNN_Q(U_i) = \sum_{(u,q),u \in U_i, q \in Q} p_q \cdot p_u \cdot RNN_q(u) \tag{2.1}$$

$RNN_q(u)$ 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.

$$RNN_q(u) = \prod_{V \in (\mathcal{U} - U_i - Q)} (1 - \sum_{v \in V, dist(u,v) < dist(u,q)} p_v) \tag{2.2}$$

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. 1.1 where the uncertain objects A, B and Q are shown. The appearance probability of each instance is also shown. e.g; appearance probability of q_1 is 0.8. According to Equation* (2.2), $RNN_{q_1}(a_1) = 1 - 0.5 - 0.5 = 0$ *because a_1 is closer to both b_1 and b_2 than q_1. Also $RNN_{q_1}(a_2) = 1$ because a_2 is closer to q_1 than both b_1 and b_2. Similarly, $RNN_{q_1}(b_1) = 1 - 0.1 - 0.9 = 0$ and $RNN_{q_1}(b_2) = 1 - 0.1 - 0.9 = 0$. According to Equation* (2.1), $RNN_Q(A) = 0 + 0.8 \times 0.9 \times 1 = 0.72$ *and $RNN_Q(B) = 0$.*

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 [5]. 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.

The table below defines the symbols and notations used throughout this paper.

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 enclosing all instances of an uncertain object U

3 Pruning Rules

Although the pruning for RNN query processing in spatial databases has been well-understood, it is *non-trivial* to devise pruning strategies for RNN query processing on uncertain data. Extension of many existing pruning rules to prune uncertain objects is either non-trivial or inefficient. For example, if we naïvely use every instance of a filtering object to perform, say, the bisector pruning [5], 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 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:

- Exploiting properties of half-spaces (Section 3.1)
- Exploiting dominance properties (Section 3.2)
- Exploiting distance properties (Section 3.3)
- Exploiting the probabilistic constraint (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. 1.1 where the bisectors between q_1 and the instances of B are drawn and the half-spaces $H_{b_1:q_1}$ and $H_{b_2:q_1}$ are shown. Intersection of the two half-spaces is shown shaded and any point that lies in the shaded area is closer to both b_1 and b_2 than q_1. For the same reason, a_1 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.

Pruning using R_{fil} and an instance q

First we present the intuition. Consider the example of Fig. 3.1 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.

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 $\bigcap_{i=1}^{2^d} H_{C_i:q}$, $dist(p,q) > maxdist(p, R_{fil})$ and thus p cannot be the RNN of q (C_i is the i^{th} corner of R_{fil}).

The pruning rule is based on Lemma 4 that is proved in Appendix (Section 9).

Consider the example of Fig. 3.2. 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.

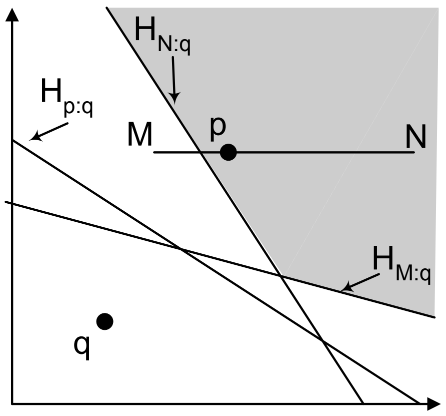


Figure 3.1: Point p lies somewhere on MN

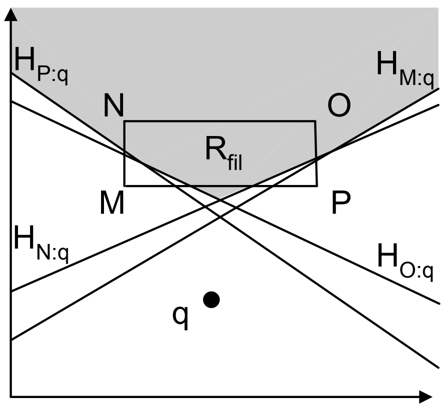


Figure 3.2: Any point in shaded area can never be RNN of q

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. 3.3 where the exact location of the query point q on line MN is not known. Unfortunately, in contrast to the previous case of Fig. 3.1, 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. 3.3 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.

One way to find the area that can be safely pruned is to move the half-spaces $H_{p:N}$ and $H_{p:M}$ such that they pass through the point c lying at the centre of the line joining p and M. Fig. 3.3 shows the point c. While $H_{p:M}$ already passes through c, we move

$H_{p:N}$ such that it passes through c (the broken line). We call such half-space that is moved to the point c a *normalized* half-space and a half-space $H_{p:N}$ that is normalized is denoted as $H'_{p:N}$. It can be proved that the intersection of the normalized half-spaces $H'_{p:M} \cap H'_{p:N}$ (the dotted area) can be safely pruned. Note that $H'_{p:M}$ is same as $H_{p:M}$ in this example because $H_{p:M}$ already passes through c.

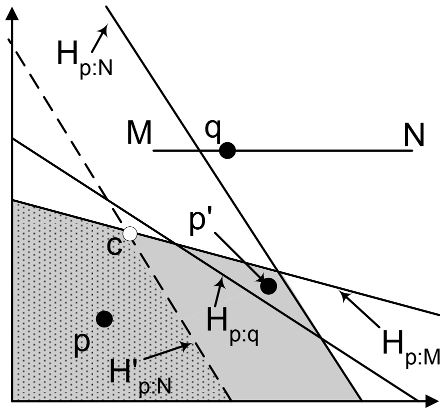


Figure 3.3: Any point in dotted-shaded area can never be RNN of q

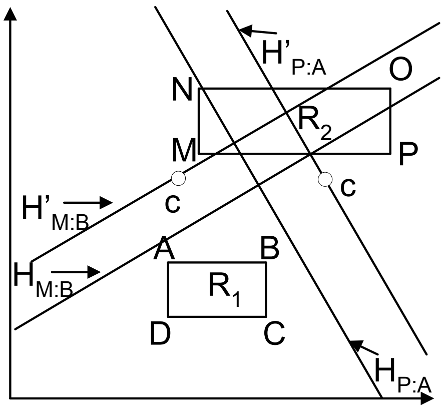


Figure 3.4: Antipodal corners and normalized half-spaces

The example of Fig. 3.3 shows the difficulty in defining the pruned area where the location of query is unknown on the line MN. This task becomes more challenging when the locations of both the query point q and data point p are not known within their respective hyper-rectangles. Before we present our pruning approach 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 corners*[1] 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. 3.4 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. 3.4 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 3.4 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 (3.1) and (3.2) define the half-space $H_{M:B}$ and its normalized half-space $H'_{M:B}$, respectively.

$$\sum_{i=1}^{d}(B[i] - M[i]) \cdot x[i] < \sum_{i=1}^{d} \frac{(B[i] - M[i])(B[i] + M[i])}{2} \tag{3.1}$$

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

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

$$\sum_{i=1}^{d}(B[i]-M[i])\times\begin{cases}\dfrac{(R1_L[i]+R2_L[i])}{2}, \text{ if } B[i]>M[i] \\ \dfrac{(R1_H[i]+R2_H[i])}{2}, \text{ otherwise}\end{cases} \tag{3.2}$$

Note that the right hand side of the Equation (3.1) cannot be smaller than the right hand side of Equation (3.2). 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 can be found in Lemma 5 in Appendix (Section 9).

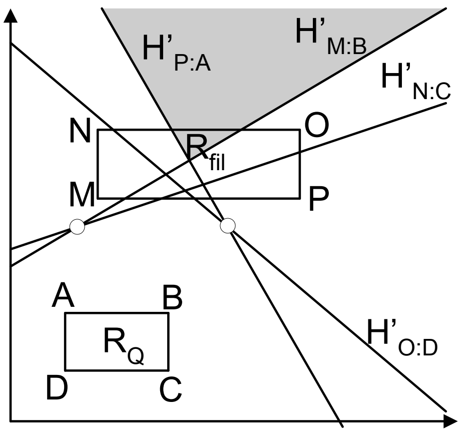


Figure 3.5: Any point in shaded area can never be RNN of any $q \in Q$

Consider the example of Fig. 3.5 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 [5].

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 [6] 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 $Rem \subseteq R_{cnd}$

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 [6]
 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 [6]
 8:   enlarge $Rem$ to enclose $Rem_i$
 9:   if $Rem = R_{cnd}$ then
10:     return $R_{cnd}$
11: return $Rem$
```

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. 3.5 and show the clipping of a hyper-rectangle R_{cnd} in Fig. 3.6. 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).

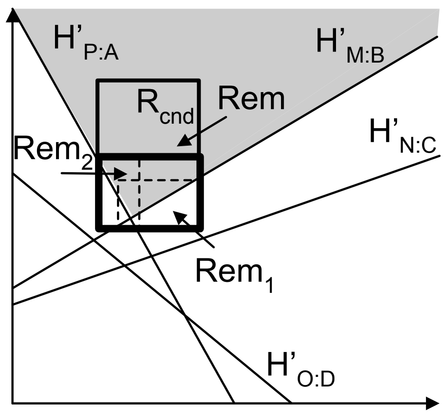


Figure 3.6: Clipping the part of R_{cnd} that can not be pruned

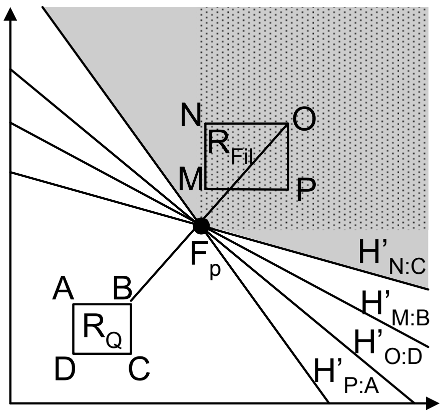


Figure 3.7: Pruning area of half-space pruning and dominance pruning

3.2 Dominance Pruning

We first give the intuition behind this pruning rule. Fig. 3.7 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[2] by R_Q in all dimensions then

[2]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.

all the normalized antipodal half-spaces meet at point F_p as shown in Fig. 3.7. 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 [3].

Let R_Q be the MBR containing instances of Q. We can obtain the 2^d regions as shown in Fig. 3.8. 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.

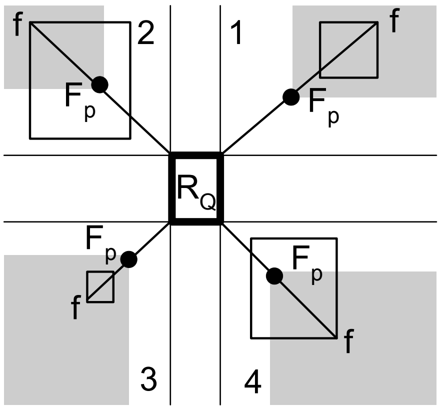


Figure 3.8: Dominance Pruning: Shaded areas can be pruned

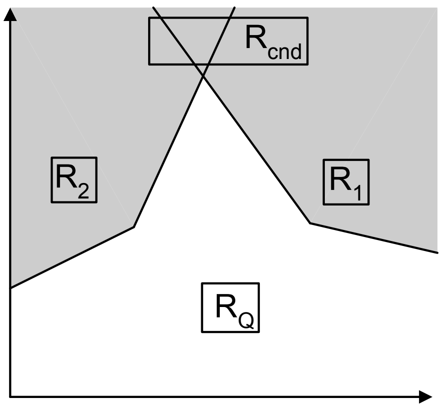


Figure 3.9: R_{cnd} can be pruned by R_1 and R_2

Fig. 3.8 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. 3.5. In general, the pruning power of this rule is less than that of the half-space pruning. Fig. 3.7 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.

3.3 Distance 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]Formal proof is given in Lemma 6 in Appendix

3.4 Probabilistic Pruning

A unique pruning rule enabled by the probabilistic constraint in our problem definition is based on estimating an upper bound of the RNN probability of candidate objects. It is made effective after parts of the candidate object have been pruned (e.g., by previous pruning rules).

PRUNING RULE 5 : Let the instances of Q be divided into n disjoint 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. We use this pruning rule in second phase of our algorithm (See Algorithm 5).

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.

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. 3.9. 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:

- For a given MBR R_{fil}, if $mindist(R_{cnd}, R_{fil}) > maxdist(R_Q, R_{cnd})$, 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 Answering Probabilistic RNN Queries

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 : 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
```

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.

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.

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).

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:        $S_{fil} = S_{fil} \cup \{c\}$
```

4.2 Refinement

In this phase, 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).

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
```

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}, $MaxProb$) (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, for each child c of e, an entry $(c, Rem, (P^c.P^{Rem}))$ is inserted into the heap. 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 (2.2)). 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.

Finding the 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}. Consider the example of Fig. 4.1 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).

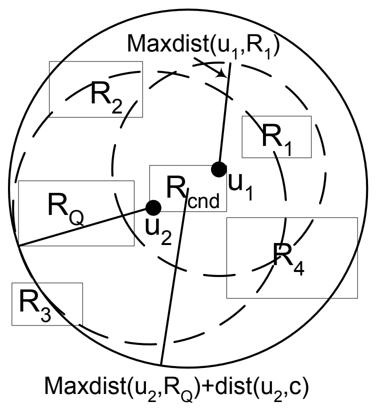


Figure 4.1: Finding the range of the global query

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. 4.1, 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.

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. 4.2. 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}.

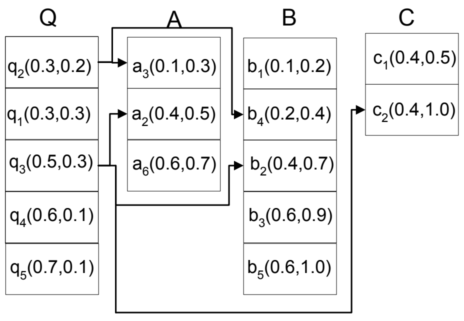


Figure 4.2: lists sorted on distance from a candidate instance u_{cnd}

Example 3: *Fig. 4.2 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 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 (2.2) 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. 4.3, 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.

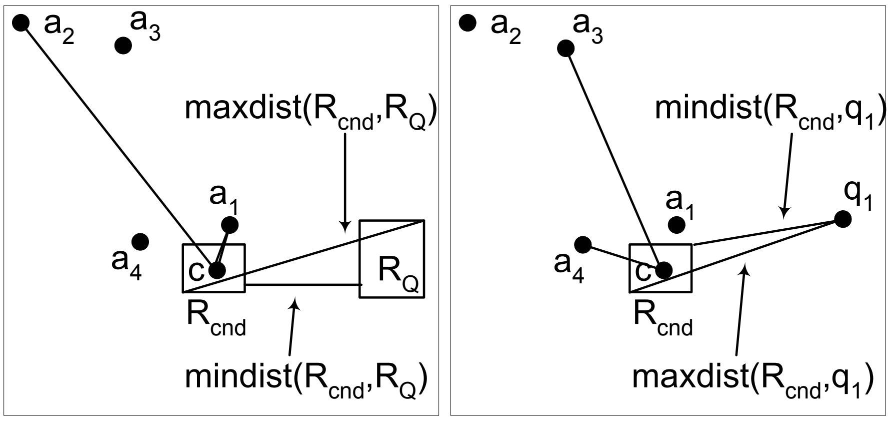


Figure 4.3: Bounding the lower and upper bound RNN probabilities

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.

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. 4.3 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). The list L_Q is sorted in ascending order of $mindist(R_{cnd}, q_i)$. 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) 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 of Q.

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 total cost. The costs shown represent the average total cost per query. We used both synthetic and real datasets.

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

The table above 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 [7], the query object follows same distribution as the underlying dataset.

The real dataset[1] 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; if $mindist(R_{cnd}, R_Q) > maxdist(R_{cnd}, R_{fil})$ then the object R_{cnd} can be pruned). 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 (2.2). Finally, the Equation (2.1) 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

[1]http://www.census.gov/geo/www/tiger/

world, we compute the RNNs using TPL [5] 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.

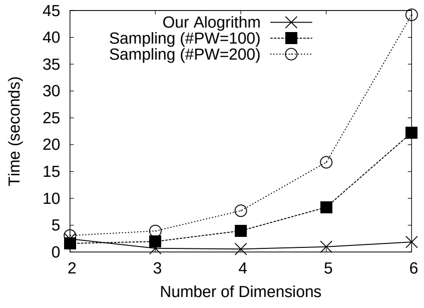


Figure 5.1: Overall cost

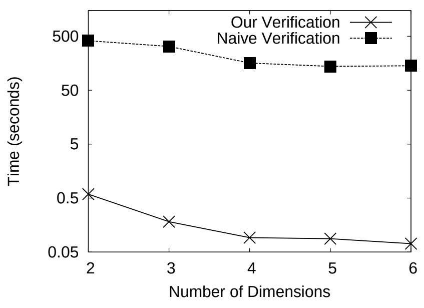


Figure 5.2: Verification cost

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. 5.2.

Fig. 5.1 compares our approach with the sampling based approximate approach 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. 5.1 shows that our algorithm is more efficient for higher dimensions and scales 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. 5.2, 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 more than thousand times faster than the naïve verification.

5.2 Performance on real dataset and effect of data distribution

Fig. 5.3 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 (the accuracy[2] varies from 60% to 75%). Our algorithm performs better than the approximate sampling based algorithm for larger threshold.

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 [5]), 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.

[2]The accuracy is the harmonic mean of the precision and recall (also known as F-measure).

For this reason, now we focus on the performance evaluation of only our proposed algorithm.

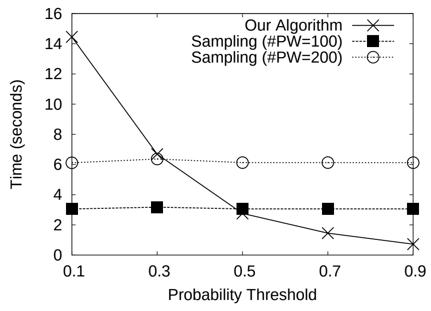


Figure 5.3: Real Dataset

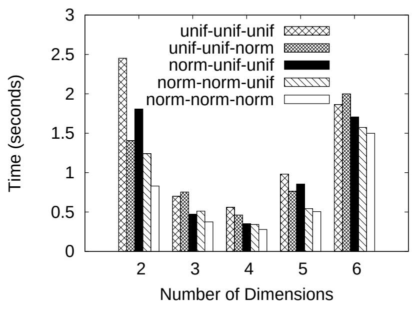


Figure 5.4: Effect of data distribution

Fig. 5.4 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. 5.4. 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. 5.5, 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. 5.6 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. 5.7 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.

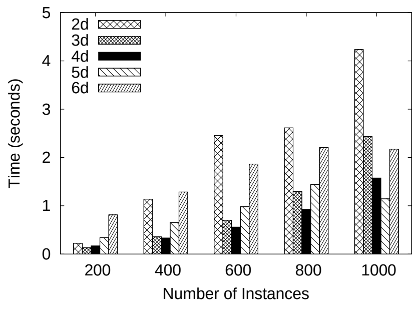


Figure 5.5: Effect of number of instances in each object

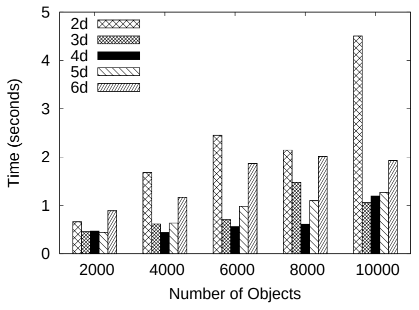


Figure 5.6: Effect of number of objects in the dataset

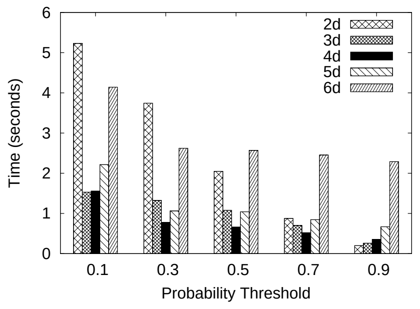


Figure 5.7: Effect of probability Threshold

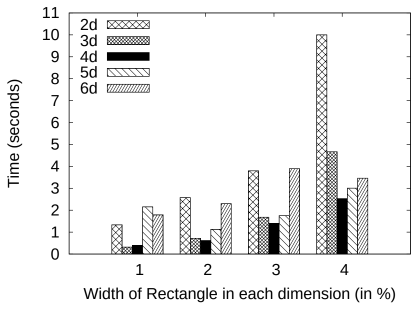


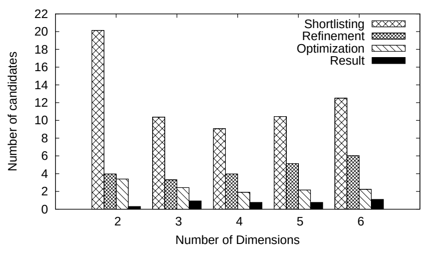
Figure 5.8: Effect of width of hyper-rectangles

In Fig. 5.8, 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.

5.5 Effectiveness and efficiency 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. 5.9 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. 5.10 shows the time taken by each of the pruning phase. Note that the 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.



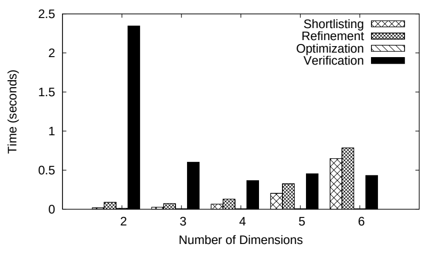
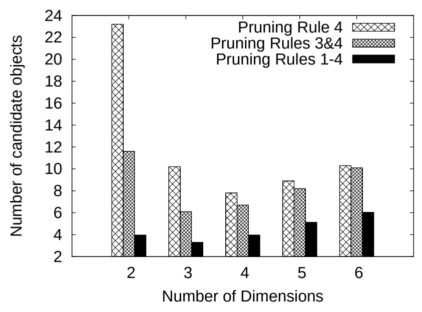


Figure 5.9: Number of candidates in S_{cnd} after each phase

Figure 5.10: Computational time taken by each phase

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. 5.9 by comparing the number of objects after *shortlisting* and *refinement* phases.



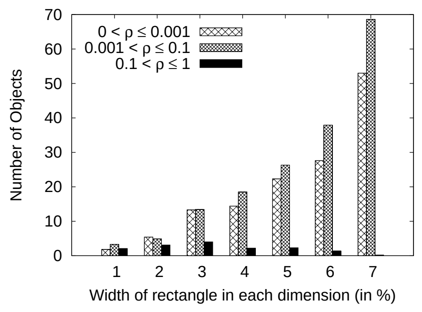


Figure 5.11: Effectiveness of pruning rules

Figure 5.12: Effect of width of hyper-rectangles

Fig. 5.11 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. The Fig. 5.11 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. 5.12 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 [1], The ORION project [8] and the references therein). Query processing on uncertain databases has gained significant attention in last few years especially in spatio-temporal databases.

In [9], the authors develop index structures to querying uncertain interval effectively. They are the first to study probabilistic range queries. In [7], 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 [10] 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 [11]. 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 [12] alongwith two effective algorithms to answer probabilistic skyline queries. While nearest neighbor queries on uncertain objects are studied in [3], 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* [13] 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 [14] 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.* [16] 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.* [5] 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* [17] 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 [18, 19, 20] and [17]. Reverse nearest neighbors in metric spaces ([21, 22] and [23]), large graphs [24] and ad hoc subspaces [25] has also been explored.

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.

Bibliography

[1] J. Widom, "Trio: A system for integrated management of data, accuracy, and lineage," in *CIDR*, 2005, pp. 262–276.

[2] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *SIGMOD Conference*, 2003, pp. 551–562.

[3] H.-P. Kriegel, P. Kunath, and M. Renz, "Probabilistic nearest-neighbor query on uncertain objects," in *DASFAA*, 2007, pp. 337–348.

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

[5] 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*. VLDB Endowment, 2004, pp. 744–755.

[6] 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*. New York, NY, USA: ACM, 1997, pp. 257–267.

[7] 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*, 2005, pp. 922–933.

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

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

[10] H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz, "Probabilistic similarity join on uncertain data," in *DASFAA*, 2006, pp. 295–309.

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

[12] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *VLDB*, 2007, pp. 15–26.

[13] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," in *SIGMOD Conference*, 2000, pp. 201–212.

[14] 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*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 485–492.

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

[16] A. Singh, H. Ferhatosmanoglu, and A. S. Tosun, “High dimensional reverse nearest neighbor queries,” in *CIKM*, 2003, pp. 91–98.

[17] W. Wu, F. Yang, C. Y. Chan, and K.-L. Tan, “Continuous reverse k-nearest-neighbor monitoring,” in *MDM*, 2008, pp. 132–139.

[18] R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis, “Nearest neighbor and reverse nearest neighbor queries for moving objects,” in *IDEAS*, 2002, pp. 44–53.

[19] T. Xia and D. Zhang, “Continuous reverse nearest neighbor monitoring,” in *ICDE*, 2006, p. 77.

[20] J. M. Kang, M. F. Mokbel, S. Shekhar, T. Xia, and D. Zhang, “Continuous evaluation of monochromatic and bichromatic reverse nearest neighbors,” in *ICDE*, 2007, pp. 806–815.

[21] 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*, 2006, pp. 515–526.

[22] 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*, 2006, pp. 788–789.

[23] Y. Tao, M. L. Yiu, and N. Mamoulis, “Reverse nearest neighbor search in metric spaces,” *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 9, pp. 1239–1252, 2006.

[24] M. L. Yiu, D. Papadias, N. Mamoulis, and Y. Tao, “Reverse nearest neighbors in large graphs,” in *ICDE*, 2005, pp. 186–187.

[25] M. L. Yiu and N. Mamoulis, “Reverse nearest neighbors search in ad hoc subspaces,” *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 412–426, 2007.

8 Glossary

Antipodal Corners: Let C be a corner of rectangle $R1$ and C' be a corner in $R2$, the two corners are called *antipodal corners* 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 8.1 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 8.1 shows two antipodal half-spaces $H_{M:B}$ and $H_{P:A}$.

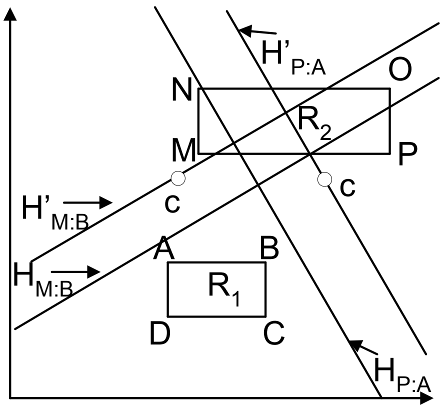


Figure 8.1: Antipodal corners and half-spaces

Normalized Half-Space: Let M and B be two points in hyper-rectangles R and Q, respectively. The normalized half-space $H'_{M:B}$ is a space defined by a bisector between M and B that passes through a point c such that $c[i] = (Q_L[i] + R_L[i])/2$ for all dimensions i for which $B[i] > M[i]$ and $c[j] = (Q_H[i] + R_H[j])/2$ for all dimensions j for which $B[j] \leq M[j]$. Fig 8.1 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 (8.1) and (8.2) define the half-space $H_{M:B}$ and its normalized half-space $H'_{M:B}$, respectively.

$$\sum_{i=1}^{d}(B[i] - M[i]) \cdot x[i] < \sum_{i=1}^{d} \frac{(B[i] - M[i])(B[i] + M[i])}{2} \tag{8.1}$$

$$\begin{aligned} &\sum_{i=1}^{d}(B[i] - M[i]) \cdot x[i] < \\ &\sum_{i=1}^{d}(B[i] - M[i]) \times \left\{ \begin{array}{lr} \dfrac{(Q_L[i] + R_L[i])}{2} & \text{if } B[i] > M[i]) \\ \dfrac{(Q_H[i] + R_H[i])}{2} & \text{otherwise} \end{array} \right\} \end{aligned} \tag{8.2}$$

Note that the right hand side of the inequality (8.1) can never be smaller than the right hand side of inequality (8.2) because M and B both lie in hyper-rectangles R and Q, respectively. For this reason $H'_{M:B} \subseteq H_{M:B}$.

Set of More Expressive Half-Spaces: A set of half-spaces $S_1 = \{H_{i:q}, ..., H_{n:q}\}$ is more expressive than any other half-space $H_{j:q}$ if it holds that $\cap_{x=i}^{n} H_{x:q} \subseteq H_{j:q}$. Note that if S_1 is a set of more expressive half-spaces then $\cap_{x=i}^{n} H_{x:q} \cap H_{j:q} = \cap_{x=i}^{n} H_{x:q}$. For example, the set of half-spaces $\{H_{M:q}, H_{N:q}\}$ in Fig. 9.1 is more expressive than the half-space $H_{L:q}$ and the shaded area is $H_{M:q} \cap H_{N:q} \cap H_{L:q} = H_{M:q} \cap H_{N:q}$.

9 Appendix: Proofs

LEMMA 1 : Let there be two subspaces SP_1 and SP_2;

$$SP_1 \Rightarrow y < Ax + B \tag{9.1}$$

$$SP_2 \Rightarrow y < Cx + D \tag{9.2}$$

where x and y are variables and A, B, C and D are constants. Both the subspaces intersect each other at $x = I_x = \frac{D-B}{A-C}$. If the whole space is partitioned into two partitions Pn_1 and Pn_2 such that Pn_1 contains all the points for which $x \geq I_x$ and Pn_2 contains all the points where $x \leq I_x$. Then we can say;

$$\left\{ \begin{array}{ll} \left\{ \begin{array}{ll} SP_1 \subseteq SP_2; & \text{in } Pn_1 \\ AND & \\ SP_2 \subseteq SP_1; & \text{in } Pn_2 \end{array} \right\} & if \;\; C > A \\ \left\{ \begin{array}{ll} SP_2 \subseteq SP_1; & \text{in } Pn_1 \\ AND & \\ SP_1 \subseteq SP_2; & \text{in } Pn_2 \end{array} \right\} & otherwise \end{array} \right\}$$

Proof We prove the case when $C > A$ and the proof of the other case is similar. Note that for $x = I_x$, the right hand sides of both the inequalities (9.1) and (9.2) would be equal and for $x > I_x$ the right hand side of the inequality (9.2) is greater than right hand side of inequality (9.1) because $C > A$. This means every point that lies in Pn_1 and satisfies inequality (9.1) would also satisfy the inequality (9.2). Hence $SP_1 \subseteq SP_2$ in space where $x \geq I_x$. Similarly, it can be proved that $SP_2 \subseteq SP_1$ in space where $x \leq I_x$. Also the proof for the case when $C \leq A$ is similar.

LEMMA 2 : Let there be three half-spaces SP_1, SP_2 and SP_3 defined by the following inequalities;

$$SP_1 \Rightarrow y < Ax + B \tag{9.3}$$

$$SP_2 \Rightarrow y < Cx + D \tag{9.4}$$

$$SP_3 \Rightarrow y < Ex + F \tag{9.5}$$

where x and y are variables and A, B, C, D, E and F are constants. The set of half-spaces $\{SP_1, SP_2\}$ is always more expressive[1] than SP_3 if both of the following are true;

1. $A > E > C$
2. $\frac{F-B}{A-E} \geq \frac{D-F}{E-C}$

Proof Since $A > E > C$, we can obtain from Lemma 1;

$$SP_2 \subseteq SP_3; \quad if\; x \geq \frac{D-F}{E-C} \tag{9.6}$$

$$SP_3 \subseteq SP_2; \quad if\; x \leq \frac{D-F}{E-C} \tag{9.7}$$

[1]The set of more expressive half-spaces is defined in Glossary (Section 8).

$$SP_3 \subseteq SP_1; \quad if\, x \geq \frac{F-B}{A-E} \tag{9.8}$$

$$SP_1 \subseteq SP_3; \quad if\, x \leq \frac{F-B}{A-E} \tag{9.9}$$

Since $\frac{F-B}{A-E} \geq \frac{D-F}{E-C}$, we obtain by joining the inequalities (9.6) and (9.8);

$$SP_2 \subseteq SP_3 \subseteq SP_1; \quad if\, x \geq \frac{F-B}{A-E} \tag{9.10}$$

From inequalities (9.10) and (9.9), it can be noted that in the whole space SP_3 is either a superset of SP_1 or SP_2. Hence $SP_1 \cap SP_2 \subseteq SP_2$.

LEMMA 3 : Let M and N be two points in d-dimensional space such that $M[i] = N[i]$ for all except one dimension j. Let q be a query point and MN be the line joining the points M and N. The set of half-spaces $\{H_{M:q}, H_{N:q}\}$ is more expressive than any $H_{L:q}$ where L is any point on the line segment MN. Fig. 9.1 shows the line and half-spaces in $2d$ space.

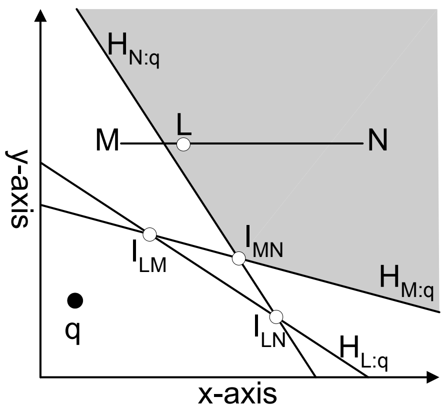


Figure 9.1: Lemma 3 in 2d-space

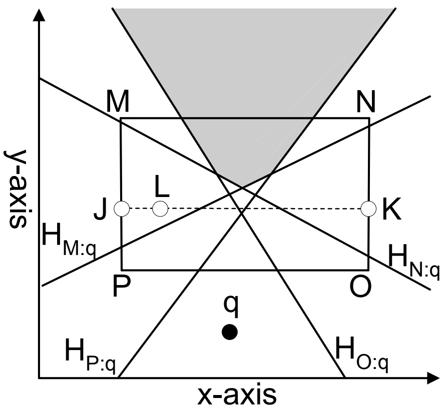


Figure 9.2: Lemma 4 in 2d-space

Proof The half-subspace $H_{N:q}$ and $H_{M:q}$ are defined by inequality (9.11) and inequality (9.12), respectively;

$$\sum_{i=1, i\neq j}^{d} (q[i] - N[i]) \cdot x[i] < (N[j] - q[j]) \cdot x[j] + \sum_{i=1}^{d} (q[i]^2 - N[i]^2)/2 \tag{9.11}$$

$$\sum_{i=1, i\neq j}^{d} (q[i] - M[i]) \cdot x[i] < (M[j] - q[j]) \cdot x[j] + \sum_{i=1}^{d} (q[i]^2 - M[i]^2)/2 \tag{9.12}$$

Let $A = (N[j] - q[j])$, $B = \sum_{i=1}^{d}(q[i]^2 - N[i]^2)/2$, $C = (M[j] - q[j])$ and $D = \sum_{i=1}^{d}(q[i]^2 - M[i]^2)/2$ be constants and $y = \sum_{i=1, i\neq j}^{d}(q[i] - M[i]) \cdot x[i]$

be a variable. Note that $M[i] = N[i]$ for all except j^{th} dimension, so we can write inequalities (9.11) and (9.12) as;

$$H_{N:q} \Rightarrow y < A \cdot x[j] + B \tag{9.13}$$

$$H_{M:q} \Rightarrow y < C \cdot x[j] + D \tag{9.14}$$

For any point L on the line MN, let $E = (L[j] - q[j])$ and $F = \sum_{i=1}^{d}(q[i]^2 - L[i]^2)/2$ be a constant. Then $H_{L:q}$ is represented by the inequality (9.15);

$$H_{L:q} \Rightarrow y < E \cdot x[j] + F \tag{9.15}$$

Without loss of generality, if we assume $M < L < N$ then $A > E > C$. Since $M[i] = N[i] = L[i]$ for all except j^{th} dimension, we calculate $\frac{F-B}{A-E}$ and $\frac{D-F}{E-C}$ which are $(N[j] + L[j])/2$ and $(M[j] + L[j])/2$, respectively. Since $\frac{F-B}{A-E} > \frac{D-F}{E-C}$, it is proved from Lemma 2 that the set of half-spaces $\{H_{M:q}, H_{N:q}\}$ is more expressive than any $H_{L:q}$.

LEMMA 4 : Let q be a query point, R be a hyper-rectangle in d-dimensional space and $\{C_1, C_2, ..., C_{2^d}\}$ be its corners. The set of half-spaces $\{H_{C_1:q}, H_{C_2:q}, ..., H_{C_{2^d}:q}\}$ is more expressive than every other half-space $H_{L:q}$ where L is any point in the hyper-rectangle R.

Proof We present the proof for a $2d$-rectangle and it can be extended to prove the Lemma for high-dimensional hyper-rectangles. In Fig. 9.2, a rectangle has been shown with four corners M, N, O and P. Note that for every point L in rectangle there exist two points J and K on the boundary of rectangle such that $\{H_{J:q}, H_{K:q}\}$ is more expressive than $H_{L:q}$ (Lemma 3). For the same reasoning, note that $\{H_{N:q}, H_{O:q}\}$ is more expressive than $H_{K:q}$ and $\{H_{M:q}, H_{P:q}\}$ is more expressive than $H_{J:q}$. Hence $\{H_{M:q}, H_{N:q}, H_{O:q}, H_{P:q}\}$ is a set of more expressive half-spaces than every half-space $H_{L:q}$. It is easy to see that this reasoning can be extended to prove the Lemma for hyper-rectangles in higher-dimensions.

LEMMA 5 : Let there be two d dimensional hyper-rectangles Q and R. The set of normalized half-spaces $\{H'_{C_1:C'_1}, ..., H'_{C_{2^d}:C'_{2^d}}\}$ is more expressive than any half-space $H_{M:N}$ where C_i is i^{th} corner of R and C'_i is its antipodal corner in Q, M is any point in hyper-rectangle R and N is any point in hyper-rectangle Q.

Proof If we prove that the set of normalized half-spaces $\{H'_{C_1:C'_1}, ..., H'_{C_{2^d}:C'_{2^d}}\}$ is more expressive than any normalized half-space $H'_{M:N}$, we can say that it is more expressive than the half-space $H_{M:N}$ because $H'_{M:N} \subseteq H_{M:N}$ by the definition of normalized half-spaces.

Unless the two points M and N are antipodal corners, it holds true that there exist two points Y and Z in R and Q, respectively, such that for all dimensions i except j, $Y[i] = M[i]$ and $Z[i] = N[i]$ and for dimension j at least one of the following two holds true;

Case 1: $(Y[j] = R_H[j]) > M[j]$ and $(Z[j] = Q_H[j]) > N[j]$

Case 2: $(Y[j] = R_L[j]) < M[j]$ and $(Z[j] = Q_L[j]) < N[j]$

We present the proof for case 1 and the proof for case 2 is similar. Let A, B, C, D, E, F and G be constants and y be variable defined as;

$$y = \sum_{i=1, i \neq j}^{d} (N[i] - M[i]) \cdot x[i]$$
$$A = Y[j] - N[j] = R_H[j] - N[j]$$
$$C = M[j] - Z[j] = M[j] - Q_H[j]$$
$$E = M[j] - N[j]$$

$$G = \sum_{i=1, i \neq j}^{d} \frac{N[i] - M[i]}{2} \times \left\{ \begin{array}{ll} (Q_L[i] + R_L[i]); & \text{if } N[i] > M[i] \\ (Q_H[i] + R_H[i]); & \text{otherwise} \end{array} \right\}$$

$$B = G + \frac{N[j] - Y[j]}{2} \times \left\{ \begin{array}{ll} (Q_L[j] + R_L[j]); & \text{if } N[j] > Y[j] \\ (Q_H[j] + R_H[j]); & \text{otherwise} \end{array} \right\}$$

$$D = G + \frac{Z[j] - M[j]}{2} \times \left\{ \begin{array}{ll} (Q_L[j] + R_L[j]); & \text{if } Z[j] > M[j] \\ (Q_H[j] + R_H[j]); & \text{otherwise} \end{array} \right\}$$

$$F = G + \frac{N[j] - M[j]}{2} \times \left\{ \begin{array}{ll} (Q_L[j] + R_L[j]); & \text{if } N[j] > M[j] \\ (Q_H[j] + R_H[j]); & \text{otherwise} \end{array} \right\}$$

The normalized half-spaces $H'_{Y:N}, H'_{M:Z}$ and $H'_{M:N}$ are defined by the following inequalities.

$$H'_{Y:N} \Rightarrow y < A \cdot x[j] + B \tag{9.16}$$
$$H'_{M:Z} \Rightarrow y < C \cdot x[j] + D \tag{9.17}$$
$$H'_{M:N} \Rightarrow y < E \cdot x[j] + F \tag{9.18}$$

According to the Lemma 2, if $A > E > C$ and $\frac{F-B}{A-E} \geq \frac{D-F}{E-C}$ then the set of normalized half-spaces $\{H'_{Y:N}, H'_{M:Z}\}$ is more expressive than the normalized half-space $H'_{M:N}$. It is easy to observe that $A > E > C$ now we compute $\frac{F-B}{A-E}$ and $\frac{D-F}{E-C}$. There are two possibilities.

Possibility 1: $N[j] \leq M[j]$; In this case $N[j]$ is always less than $Y[j]$ and $\frac{F-B}{A-E} = \frac{(Q_H[i]+R_H[i])}{2}$. On the other hand $Z[j]$ might be greater, lesser or equal to $M[j]$. To maximize $\frac{D-F}{E-C}$, we assume that $Z[j] > M[j]$ and compute $\frac{D-F}{E-C} = \frac{(Q_L[i]+R_L[i])}{2}$. Hence $\frac{F-B}{A-E} > \frac{D-F}{E-C}$.

Possibility 2:$N[j] > M[j]$; In this case $Z[j]$ is always greater than $M[j]$. We can compute that $\frac{D-F}{E-C} = \frac{(Q_L[i]+R_L[i])}{2}$. On the other hand $N[j]$ might be greater, lesser or equal to $Y[j]$. To minimize $\frac{F-B}{A-E}$, we assume that $N[j] > Y[j]$ and compute $\frac{F-B}{A-E} = \frac{(Q_L[i]+R_L[i])}{2}$. Hence $\frac{F-B}{A-E} \geq \frac{D-F}{E-C}$.

We have proved that the set of normalized half-spaces $\{H'_{M:Z}, H'_{Y:N}\}$ is more expressive than the normalized half-space $H'_{M:N}$. It can be found that for any such $H'_{M:Z}$ (or $H'_{Y:N}$), there exists a set of normalized half-spaces that is more expressive unless M and Z (or Y and N) are two antipodal corners. Hence the set of antipodal normalized half-spaces $\{H'_{C_1:C'_1}, ..., H'_{C_{2^d}:C'_{2^d}}\}$ is more expressive than any other normalized half-space $H'_{M:N}$ where M and N are the points in hyper-rectangle R and Q, respectively. Since $H'_{M:N} \subseteq H_{M:N}$, we can say that the set $\{H'_{C_1:C'_1}, ..., H'_{C_{2^d}:C'_{2^d}}\}$ is more expressive than any half-space $H_{M:N}$. This completes the proof.

LEMMA 6 : Let Q and R be two hyper-rectangles in d dimensional space such that for every dimension i, either $R_H[i] \leq Q_L[i]$ or $Q_H[i] \leq R_L[i]$ and for at least one dimension j either $R_H[j] < Q_L[j]$ or $Q_H[j] < R_L[j]$ (i.e; there exists a dominance relationship such that R is dominated by Q). Let F_p and p be two points such that $p > (F_p[i] = (Q_H[i] + R_H[i])/2)$ for any dimension i for which $Q_H[i] \leq R_L[i]$ and $p < (F_p[j] = (Q_L[j] + R_L[j])/2)$ for any dimension j for which $R_H[j] \leq Q_L[j]$ (i.e; p is dominated by F_p in the same way as R is dominated by Q). Then we can say $maxdist(p, R) > mindist(p, Q)$.

Proof We can prove the lemma by showing that the point p lies in every normalized half-space $H'_{M:N}$ where M is a point in R and N is a point in Q. The normalized half-space can be defined as;

$$\begin{aligned}
&\sum_{i=1}^{d}(N[i] - M[i]) \cdot x[i] < \\
&\sum_{i=1}^{d}(N[i] - M[i]) \times \left\{ \begin{array}{lr} \dfrac{(Q_L[i] + R_L[i])}{2} & \text{if } N[i] > M[i]) \\ \dfrac{(Q_H[i] + R_H[i])}{2} & \text{otherwise} \end{array} \right\}
\end{aligned} \tag{9.19}$$

We evaluate the left hand side of the inequality (9.19) w.r.t F_p (e.g; $x[i] = F_p[i]$);

$$\sum_{i=1}^{d}(N[i] - M[i]) \times \left\{ \begin{array}{ll} \dfrac{Q_L[i] + R_L[i])}{2} & \text{if } Q_L[i] \geq R_H[i] \\ \dfrac{Q_H[i] + R_H[i])}{2} & \text{if } R_L[i] \geq Q_H[i] \end{array} \right\} \tag{9.20}$$

It can be observed that the value in (9.20) is always equal to the RHS of the inequality (9.19) because M is a point in R and N is a point in Q. So for any dimension i where $Q_L[i] \geq R_H[i]$, $N[i] - M[i]$ is always positive. Similarly, for any dimension j for which $R_L[i] \geq Q_H[i]$, $N[i] - M[i]$ is always negative.

Furthermore, it can be noted by the definition of the point p that the LHS of the inequality (9.19) when evaluated w.r.t p is always less than what we obtained in (9.20). Hence p lies in every normalized half-space $H'_{M:N}$.