Efficient Probabilistic Reverse Nearest Neighbor Query Processing on Uncertain Data

Thomas Bernecker, Tobias Emrich, Hans-Peter Kriegel
Matthias Renz, Stefan Zankl, Andreas Züfle
Department of Computer Science, Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538 Munich, Germany
{bernecker,emrich,kriegel,renz,zankl,zuefle}@dbs.ifi.lmu.de

ABSTRACT

Given a query object q, a reverse nearest neighbor (RNN) query in a common certain database returns the objects having q as their nearest neighbor. A new challenge for databases is dealing with uncertain objects. In this paper we consider probabilistic reverse nearest neighbor (PRNN) queries, which return the uncertain objects having the query object as nearest neighbor with a sufficiently high probability. We propose an algorithm for efficiently answering PRNN queries using new pruning mechanisms taking distance dependencies into account. We compare our algorithm to state-of-the-art approaches recently proposed. Our experimental evaluation shows that our approach is able to significantly outperform previous approaches. In addition, we show how our approach can easily be extended to PRkNN (where $k > 1$) query processing for which there is currently no efficient solution.

1. INTRODUCTION

A Reverse k-Nearest Neighbor (RkNN) query retrieves all objects having a given query object as one of their k nearest neighbors. The RkNN query processing has been studied extensively on certain data [1, 2, 3]. However, due to the immense number of applications dealing with uncertain data, novel solutions to cope with uncertain objects are required. The main challenge here is that the event that an object belongs to an RkNN result set, is no longer a predicate, but a random variable that may be true with some probability. In this paper, we study the problem of probabilistic reverse k-nearest neighbor (PRkNN) search in uncertain databases. A PRkNN query returns the set of objects having a sufficiently high probability to be the reverse k-nearest neighbor of a query object. Let us note that the query object can be uncertain as well. Traditional methods as proposed in [2, 3] do not qualify for uncertain objects, since an uncertain object – instead of being a single point in a multi-dimensional space – is a random variable defined by the probability distribution over the distance space. According to the possible worlds model [4], an uncertain database can be viewed as a set of possible database instances (worlds), to which traditional pruning methods can be applied to. Since the number of possible worlds is exponential in the number of objects, we need special methods to avoid consideration of all possible worlds.

Proceedings of the VLDB Endowment, Vol. X, No. Y

UserID	Movie	Humor	Action
134	The Life of Brian	93 %	5 %
...	...	...	...
256	The Life of Brian	12 %	10 %
...	...	...	...

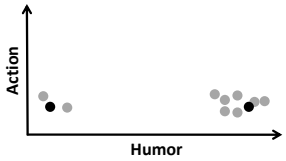
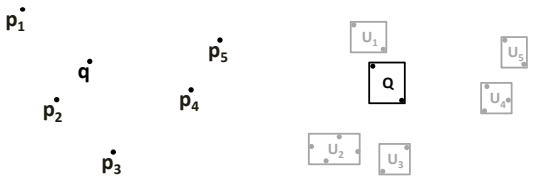


Figure 1: Uncertain object example: user ratings.

There is a wide field of applications for PRNN queries (k=1), e.g. decision support, marketing, location-based services among others [5, 6]. For instance, consider a movie recommendation system that reports a list of movies that are similar to other movies that a user likes. What a user likes is however a very subjective variable that depends on user-specific preferences which cannot be measured. Therefore, each movie record is assumed to be associated with a set of user reviews, each consisting of a set of attribute values. Examples of such attributes are classification of the genre, humoristic value and suspense. An example for two such records is given in Figure 1. Thus, each movie record is represented by multiple records from different users. Differences between records of the same movie reflect the uncertainty in the user ratings. The advantage of using this uncertain data instead of simply using the average user recommendation of each movie record is shown by the following example: Consider a movie like "Monty Python's The Life of Brian" [7]: Many users will rate this movie as extremely funny. However, since this movie is based on a rather black sense of humor, some users may rate this movie as absolutely not funny. Thus, this movie would have an average of "moderately funny" and would result in a very large distance to other funny movies. Thus, this movie would never be recommended to users purchasing funny movies, even though there is a high probability that a user looking for funny movies may indeed be interested in this movie.

In this paper, we first propose novel efficient methods for the PRNN (PRkNN with k = 1) query that outperforms the latest state-of-the-art solutions and then show how this approach can be extended to efficiently answer PRkNN queries (for $k \geq 1$) as a first solution of this problem. The contributions of this paper can be summarized as follows:

- We propose a general framework for PRNN query algorithms and show how the two state-of-the-art approaches fit into it.
- By means of a recently proposed spatial pruning criterion [8] we derive an efficient probabilistic pruning filter criterion. This criterion is shown to be correct in accordance the *possible worlds* semantics as it treats inherent distance dependencies in a correct way.



(a) RNN example. (b) PRNN example.

Figure 2: Examples for RNN and PRNN.

- We show how the new techniques for spatial pruning, probabilistic pruning and verification are combined to obtain an efficient PRNN algorithm. Additionally we show how this algorithm can be extended in order to answer PRkNN queries, which is the first solution to this problem.
- For the case where the objects are given by a discrete uncertainty model, we will show that all proposed techniques are correct and the algorithm efficiently yields the exact result. In the continuous case, where verification cannot be performed efficiently, the algorithm is able to approximate the exact result as tight as needed and returns the error bounds.
- We experimentally show that our proposed algorithm performs better than the two existing solutions under various settings. To the best of our knowledge, this is the first comparison of the two existing PRNN solutions.

The rest of this paper is organized as follows: First we formally define the problem of PRNN queries in Section 2. In Section 3, we describe the framework for PRNN query processing. In Section 4, we introduce novel spatial and probabilistic pruning filter criteria on uncertain data. In Section 5 we show how to extend the PRNN framework and algorithm to efficiently answer PRkNN queries. All proposed techniques are experimentally evaluated in Section 6. Finally Sections 7 and 8 review related work and conclude this paper.

2. PROBLEM DEFINITION

In this section, we give a formal definition of the Probabilistic Reverse Nearest Neighbor (PRNN) problem in uncertain databases. Therefore, we briefly review conventional reverse nearest neighbor queries and the uncertainty model used in this work.

DEFINITION 1. *Given a set of (certain) point objects $P \subset \mathbb{R}^d$ and a query object $q \in \mathbb{R}^d$, a reverse nearest neighbor query (RNN_q) returns all points $p \in P$ which have q as their nearest neighbor, formally:*

$$RNN_q = \{p | \forall p' \in P \setminus p : dist(p,q) \leq dist(p,p')\},$$

where $dist()$ is the Euclidean distance in $\mathbb{R}^d$.

An example is illustrated in Figure 2(a). Consider the point object set ($p_{1-5} \in P$) with q as query object. Here, the result of an RNN query would contain the objects p_1 and p_2.

2.1 Uncertainty Model

Following the Block-Independent Disjoint Scheme ([9]) which is one of the most commonly used uncertainty models, we assume the following: a probabilistic database $\mathcal{D}$ is given by a set of uncertain objects $\mathcal{D} = \{U_1, \ldots, U_n\}$ with d uncertain attributes. An uncertain object U_i is represented by a set of d-dimensional points $u_1, \ldots, u_m$ reflecting all possible instances of U_i. Each instance u_j is assigned with a probability $P(u_j)$ denoting the probability that U_i appears at u_j, i.e. all instances of U_i reflect the probability distribution of U_i. The probability distributions of each two objects are pairwise independent and the events of occurrence of all instances $u \in U_i$ are mutually exclusive. For clarity we assume $P(U_i) = \Sigma_{j=1}^{m} u_j = 1$, although the proposed algorithms can easily be adapted to the case where $P(U_i) \leq 1$. A possible world $W = u_1, \ldots, u_n$ is a set of instances containing one instance from each object and occurring with an appearance probability of $P(W) = \Pi_{i=1}^{n} P(u_i)$. Let Ω denote the set of all possible worlds, then $\Sigma_{W \in \Omega} P(W) = 1$.

2.2 PRNN Queries in Uncertain Databases

For PRNN queries on uncertain databases the threshold parameter τ is introduced (cf. [5, 6]). Using τ as threshold, the user can restrict the result set to objects which have at least a predefined probability to be in the result in order to avoid reporting unnecessarily many results that are unlikely. A probabilistic nearest neighbor query $PRNN_Q^\tau$ then returns the set of all objects $U_i \in \mathcal{D}$ where $P(U_i \in RNN_Q)$ (in the following denoted by $P(RNN_Q(U_i))$) $\geq \tau$. Naively, this probability can be calculated by performing a (non-probabilistic) RNN query on each possible world:

$$P(RNN_Q(U_i)) = \sum_{W \in \Omega} P(W) \cdot \delta(RNN_Q^W(U_i))$$

where $\delta(RNN_Q^W(U_i))$ is an indicator function that is 1 if U_i is a reverse nearest neighbor of Q in world W and 0 otherwise. In the example given in Figure 2(b), $RNN_Q(U_1)$ holds in all possible worlds, therefore $P(RNN_Q(U_1)) = 1$. In contrast, $P(RNN_Q(U_2)) = P(RNN_Q(U_4)) = P(RNN_Q(U_5)) = 0$, since there is no possible world in which $RNN_Q(U_2)$, $RNN_Q(U_4)$ or $RNN_Q(U_5)$ hold, as in each possible world the nearest neighbor of U_2 is U_1 and the nearest neighbor of U_4 is U_5 and vice versa. For U_3, we obtain the probability $P(RNN_Q(U_3))$ by building the sum of the probabilities of all possible worlds where at least one of the objects U_i ($i \in \{1, 2, 4, 5\}$) is closer to U_3 than Q to U_3. Obviously, this brute-force approach taking each possible world into account is in general not applicable because the number of possible worlds grows exponentially with the number of involved uncertain objects.

In order to shrink down the computational overhead of such queries, in this paper we introduce efficient filter methods used to exclude (prune) as many objects as possible from the expensive query evaluation process. An object B can be pruned if we find another object A that is closer to B than the query object Q. Since the event that an uncertain object A prunes another uncertain object B with respect to an uncertain query object Q, i.e. $dist(A,B) < dist(Q,B)$, is a random variable, in the following denoted by $A \prec_Q B$, we need the concept of *probabilistic pruning*.

DEFINITION 2 (PROBABILISTIC PRUNING). *Given three uncertain objects A, B and the query object Q. The probability $P(A \prec_Q B)$ denotes the probability that A prunes B w.r.t. Q.*

Naively, we can compute $P(A \prec_Q B)$ by simply adding the probabilities of all possible worlds in which A prunes B, exploiting inter-object independency:

$$P(A \prec_Q B) = \sum_{a_i \in A} \sum_{b_j \in B} \sum_{q_k \in Q} \delta(a_i, b_j, q_k) \cdot P(a_i) \cdot P(b_j) \cdot P(q_k) \quad (1)$$

where the function $\delta(a,b,q)$ returns 1 if $dist(a,b) < dist(q,b)$ and 0 otherwise.

The problem of this naive approach is the computational cost of the triple-sum which is cubic in the number of instances of the uncertain objects. The number of instances x_i of an uncertain object X may in general be large, for example if the instances are

obtained by Monte-Carlo sampling of an unknown PDF. Another problem is that the probability $P(A \prec_Q B)$ cannot directly be used to derive the probability that an object B is the RNN of Q. For two uncertain objects A_1 and A_2, the two events $A_1 \prec_Q B$ and $A_2 \prec_Q B$ are mutually dependent because both events depend on the assumptions made for object B (more details will be found in Section 4.3.2). In order to keep correctness w.r.t. the possible worlds semantics, this problem has to be taken into account.

3. PRNN ALGORITHM SKETCH

Before introducing details of our PRNN query method, we will give a general framework for efficient PRNN query processing in an abstract fashion.

Approximation of Objects

The probability distribution (or more specifically the uncertainty region) assigned to an uncertain object can become arbitrarily complex causing expensive distance computations at query time. A common solution to overcome this problem is to use conservative approximations, like spheres or rectangles providing efficient distance computation in a filter step. For efficient processing, these approximations are often organized in a hierarchical spatial index structure like the R-tree [10].

Spatial Pruning

It is possible to (spatially) prune objects without considering their probability distributions when using only the (spatial) approximations of the objects. Therefore, a pruning technique is needed. For instance an object B can be pruned by an object A for a query Q if $MaxDist(A, B) < MinDist(B, Q)$. The used pruning technique for uncertain objects efficiently organized by an index is easily extendable for pruning higher-level pages of the index.

Probabilistic Pruning

Probabilistic pruning is performed for objects that cannot be pruned spatially. In the probabilistic pruning step, the uncertainty regions of objects are partitioned. The aim of this partitioning is to prune more objects based on the probability threshold τ.

Verification

An object U_i which cannot be pruned by the pruning techniques is denoted as candidate. The next step requires each candidate to be verified, which means it has to be checked if $P(RNN_Q(U_i)) \geq \tau$. This involves finding all objects which affect this probability and considering these objects in more detail. The verification step is very expensive, since many possibilities have to be considered.

4. HIERARCHICAL PRNN PROCESSING

In this section, we propose our approach for PRNN processing implementing the above framework in consideration of the discrete uncertainty model. For a comparison with state-of-the-art solutions, in Appendix A we show how the two existing solutions by Cheema et al. [5] (in the following called CLWZP) and by Lian et al. [6] (in the following called LC) are implemented according to this framework. The complete algorithm of our PRNN query processing approach can be found in Appendix D and in Appendix E we show how the proposed approaches can be extended to continuous uncertainty models.

4.1 Approximation

Similar to the approximation technique used for the CLWZP algorithm ([5]), in order to approximate uncertain objects we use minimum bounding boxes covering the uncertainty regions. This approximation is mainly used for spatial pruning. For probabilistic pruning, we need a more detailed object approximation. Therefore, we additionally assume that each uncertainty region of an object $X \in \mathcal{D} \cup \{Q\}$ is hierarchically decomposed using a hierarchical space partitioning scheme. Specifically, we use an R*-tree [11] to hierarchically organize the instances of X. In addition to the spatial keys, each index entry stores the aggregated probability of all instances in the corresponding subtree. When traversing the index assigned to an uncertain object X in a breadth-first manner, each level of the R*-tree provides a disjoint and complete partitioning $\underline{\mathcal{X}}$ of X such that each partition $X' \in \underline{\mathcal{X}}$ contains a non-empty set of $m' \leq m$ instances $\{x'_1, ..., x'_{m'}\}$ and

$$\bigcup_{X' \in \underline{\mathcal{X}}} = \{x_1, ..., x_m\} \text{ and } \forall X'_i \in \underline{\mathcal{X}}, X'_{j \neq i} \in \underline{\mathcal{X}} : X'_i \cap X'_j = \emptyset.$$

An index entry representing partition X'_i contains the aggregated probability $P(X'_i) = \sum_{x'_j \in X'_i} P(x'_j)$. Note that disjoint property implies that any instance $x \in X$ is contained in at most one partition. The rectangular approximations of the instances contained in each R*-tree node however may overlap.

Let us note that we have to carefully select the refinement resolution since the PRNN computation is CPU-bound, as we will see in our experiments (cf. Section 6). We obtain a good control of this variable by using a very low R*-tree node capacity, e.g. in our experiments we used less than four entries per node.

To summarize, we use a memory-resident R*-tree to organize the objects $X \in \mathcal{D}$ (global R*-Tree). The leaf entries containing the MBRs of the objects point to the local R*-trees of the objects. The local R*-trees are stored in a breadth-first manner, such that the highest levels of the trees can be obtained with one single scan.

4.2 Spatial Pruning

In accordance with our framework, here we propose a spatial pruning method that uses only the spatial keys of the uncertainty regions of the uncertain objects without taking into account further knowledge about the probability distribution. Our spatial pruning approach adopts the spatial domination concepts proposed in [8] in the context of certain data.

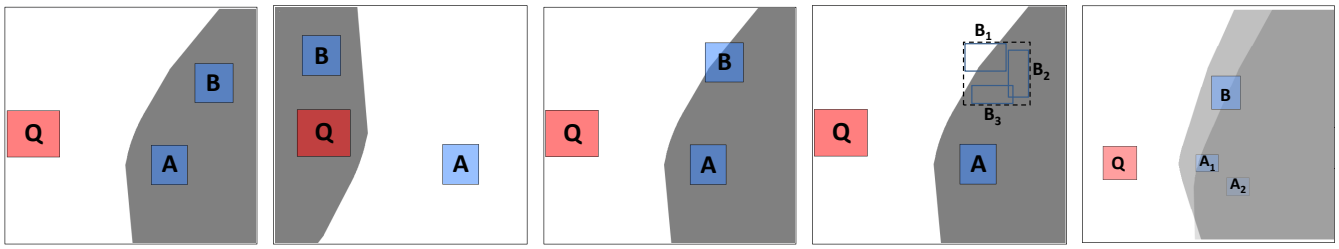
LEMMA 1 (COMPLETE PRUNING). *Let A, B, Q be uncertain objects with rectangular uncertainty regions, then:*

$$\sum_{i=1}^{d} \max_{b_i \in \{B_i^{min}, B_i^{max}\}} (\textit{MaxDist}(A_i, b_i)^2 - \textit{MinDist}(Q_i, b_i)^2) < 0 \Rightarrow P(A \prec_Q B) = 1 \quad (2)$$

where X_i ($X \in \{A, B, Q\}$) denotes the projection interval of the rectangular uncertainty region of X on the i^{th} dimension, X_i^{min} (X_i^{max}) denotes the lower (upper) bound of the interval X_i, and MaxDist(I, p) (MinDist(I, p)) denotes the maximal (minimal) distance between a one-dimensional interval I and a one-dimensional point p. In the following, we will use the notation $A \prec_Q^{\square} B$ for the predicate on the left-hand side of the implication in Equation 2.

A formal proof can be found in Appendix B.1.

An example is given in Figure 3(a), where the uncertainty regions of the uncertain objects A, B and Q are depicted. In addition, the pruning region of A is shown in grey, that is the region containing exactly the points p in space for which it holds that p is definitely closer to all points in A than to any point in Q. Note that the pruning regions do not have to be materialized, here we only use them for illustration purposes. In this example, object B



(a) $P(A \prec_Q B) = 1$ (b) $P(A \prec_Q B) = 0$ (c) $0 \leq P(A \prec_Q B) \leq 1$ (d) Decomposition of B (e) Decomposition of A

Figure 3: Visualization of different pruning techniques (a)-(c) and object decompositioning (d)-(e).

is completely contained in the pruning region of A and thus the predicate $A \prec_Q^{\square} B$ holds. Exploiting Equation 2, we can safely prune B. In fact, to decide whether B is completely contained in the pruning region we only have to apply Lemma 1.

In addition, we can use Lemma 1 to determine the objects A that cannot prune B in any possible world:

COROLLARY 1 (COMPLETE NON-PRUNING).

$$P(A \prec_Q B) = 0 \Leftrightarrow P(Q \prec_A B) = 1.$$

PROOF. The above corollary is evident, since in any world with instances $a_i \in A$, $b_j \in B$ and $q_k \in Q$, it holds that $dist(q_k, b_j) < dist(a_i, b_j) \Leftrightarrow \neg(dist(a_i, b_j) < dist(q_k, b_j))$ due to the anti-symmetry of $<$. □

This corollary allows to efficiently determine if object A cannot possibly prune B, by evaluating $Q \prec_A^{\square} B$ using Lemma 1. An example is given in Figure 3(b) where object B is completely contained in the pruning region of Q and thus, A cannot possibly prune B. Consequently, we can return any object $B \in \mathcal{D}$ as a true hit, if for all objects $A \in \mathcal{D} \setminus \{B\}$ the predicate $Q \prec_A^{\square} B$ holds.

4.3 Probabilistic Pruning

Using the techniques proposed in the previous section, we can efficiently prune any uncertain object $B \in \mathcal{D}$ for which it holds that there exists an uncertain object $A \in \mathcal{D} \setminus \{B\}$ such that $A \prec_Q^{\square} B$ holds, exploiting Lemma 1. Now, we consider the case where the probability that A prunes B is between 0 and 1. In consideration of the possible worlds semantics, that means that there exist worlds in which A prunes B, but not all possible worlds satisfy this criterion. An exemplary situation is given in Figure 3(c): here, the uncertainty region of B is not completely contained in the pruning region of A. Thus, there may exist instances $b_i \in B$ that are not located in the pruning region of A. That in turn means, that there may be instances $a_i \in A$, $b_j \in B$ and $q_k \in Q$ such that $dist(b_j, q_k) < dist(b_j, a_i)$ and, thus, $P(A \prec_Q B) < 1$.

4.3.1 Individual Object Pruning

In general, the exact computation of $P(A \prec_Q B)$ is very expensive (O(m^3)). In this section we show how to compute lower/upper bounds of $P(A \prec_Q B)$ efficiently. This allows us to quickly detect in a filter step whether the probability that B can be pruned is at least $1 - \tau$ (or cannot exceed τ) in order to prune B (or to return B as a true hit).

The key idea is to decompose the uncertainty region of an object X into subregions for which we know the probability that X is located in that subregion (cf. Section 4.1). Therefore, if $P(A \prec_Q B) < 1$, then there may still exist subregions $A' \subset A$, $B' \subset B$ and $Q' \subset Q$ such that $P(A' \prec_{Q'} B') = 1$. Examples of such situations are given in Figures 3(d) and 3(e). In Figure 3(d), some partitions of the uncertain object B are pruned by the uncertainty region of A. Given the total sum S of the probabilities of all partitions of B that cannot be pruned, we can conclude that B is an RNN of Q with a probability of at most S. In Figure 3(e), object A is divided into two partitions A_1 and A_2. It can be observed that B is fully contained in the pruning region of A_1 but not in the pruning region of A_2. Given the probability P of A_1, we can conclude that A prunes B with a probability of at least P. Thus, given a complete and disjoint object partitioning $\underline{\mathcal{A}}$, $\underline{\mathcal{B}}$ and $\underline{\mathcal{Q}}$ as described in Section 4.1, we can identify triples of subregions ($A' \in \underline{\mathcal{A}}$, $B' \in \underline{\mathcal{B}}$, $Q' \in \underline{\mathcal{Q}}$) for which $P(A' \prec_{Q'} B') = 1$ (cf. Equation 2) holds. Let $\delta(A', B', Q')$ be the following indicator function (which can be computed efficiently by using Lemma 1):

$$\delta(A', B', Q') = \begin{cases} 1, & \text{if } A' \prec_{Q'}^{\square} B' \\ 0, & \text{else} \end{cases}$$

LEMMA 2. *Let A, B and Q be uncertain objects with disjoint and complete object decompositions $\underline{\mathcal{A}}, \underline{\mathcal{B}}$ and $\underline{\mathcal{Q}}$, respectively. To derive a correct lower bound $P_{LB}(A \prec_Q B)$ of the probability $P(A \prec_Q B)$ that A prunes B, we can accumulate the probabilities of combinations of these subregions as follows:*

$$P_{LB}(A \prec_Q B) =$$

$$\sum_{A' \in \underline{\mathcal{A}}, B' \in \underline{\mathcal{B}}, Q' \in \underline{\mathcal{Q}}} P(A') \cdot P(B') \cdot P(Q') \cdot \delta(A', B', Q').$$

A formal proof can be found in Appendix B.2.

Analogously, we can define an upper bound of $P(A \prec_Q B)$ using the intuition of Corollary 1:

LEMMA 3. *An upper bound $P_{UB}(A \prec_Q B)$ of $P(A \prec_Q B)$ can be derived as follows:*

$$P_{UB}(A \prec_Q B) = 1 - P_{LB}(Q \prec_A B).$$

Following the partitioning concept proposed in Section 4.1, we can control the resolution and can refine the partitioning on demand. Naturally, the more refined the object partitions are, the tighter are the bounds that can be computed but the higher are the corresponding cost of deriving them. In particular, starting from the entire MBRs of the objects, we can progressively partition them by traversing the object instance index to iteratively derive tighter probability bounds until a desired degree of certainty is achieved (based on some threshold). In general, this allows us to significantly prune the space of candidate objects. However, the RNN probability of a given object B cannot be straightforwardly derived with the use of these bounds, as we will see in the following section.

4.3.2 Joint Object Pruning

Given the probability bounds $P_{LB}(A \prec_Q B)$ and $P_{UB}(A \prec_Q B)$, the next problem is to accumulate these probabilities to obtain an approximation of the probability $P(RNN_Q(B))$ that object B is an RNN of Q. Note that in the scope of this paper the term "approximation of a probability" means a conservative bound (consisting of an upper and lower bound) of the exact probability. The problem at issue is that, though all objects are assumed to be mutually independent, the two events $A_1 \prec_Q B$ and $A_2 \prec_Q B$ are generally mutually dependent. For example, consider a case where Q is a (certain) point and A_1 and A_2 are (certain) points, as well, having the same location. B is an uncertain object such that $P(A_1 \prec_Q B) = P(A_2 \prec_Q B) = 50\%$. Since A_1 and A_2 have the same location, it holds that $A_1 \prec_Q B \Leftrightarrow A_2 \prec_Q B$, i.e. if and only if A_1 prunes B, then A_2 prunes B. The reason for this dependency is that both random events depend on the positions of B and Q.

To avoid this problem, we present a way to conservatively approximate the probability $P(RNN_Q(B))$ while accounting for the dependencies between the random variables $A_i \prec_Q B$ ($A_i \in \mathcal{D}$).

4.3.3 Decomposition of Database Object A_i

Let $I = \{A_1, ..., A_{|I|}\}$ denote the set of *influence objects* of B, which neither completely prune B w.r.t. Q nor are completely pruned by B.[1] We only have to consider the set of random events $\{A_1 \prec_Q B, ..., A_{|I|} \prec_Q B\}$, since the objects A_i for which $P(A_i \prec_Q B) = 0$ holds do not have any influence on $P(RNN_Q(B))$, and if there exists an object A_i for which it holds that $A_i \prec_Q^{\square} B$, then we can already conclude that $P(RNN_Q(B)) = 0$. But due to the problem of mutual dependencies between pruning events (cf. Section 4.3.2), here we cannot simply use the probability bounds $P_{LB}(A_i \prec_Q B)$ and $P_{UB}(A_i \prec_Q B)$ (cf. Lemma 2) directly, as this would yield incorrect results. However, we can use the observation that the objects A_i are mutually independent and each candidate object A_i only appears in a single random variable $A_1 \prec_Q B, ..., A_{|I|} \prec_Q B$. Exploiting this observation, we can decompose[2] the objects $A_1, ..., A_{|I|}$ only to obtain mutually independent bounds for the probabilities $P(A_1 \prec_Q B), ..., P(A_{|I|} \prec_Q B)$, as stated by the following lemma:

LEMMA 4. *If B and Q are not decomposed, i.e. if $\underline{\mathcal{B}} = \{B\}$ and $\underline{\mathcal{Q}} = \{Q\}$, then $P(RNN_Q(B))$ is lower bounded by*

$$P_{LB}(RNN_Q(B)) = \prod_{A_i \in I} 1 - P_{UB}(A_i \prec_Q B).$$

PROOF. In a nutshell, the lemma can be proven by showing that the probability bounds $P_{UB}(A_i \prec_Q B), 1 \leq i \leq |I|$ can be computed independently of each other, since the computation of $A_i \prec_Q^{\square} B$ for $A_i \in \underline{\mathcal{A}}$ makes no assumptions on the positions of objects B and Q. This independence can be shown by exploiting that the bounds $P_{UB}(A_i \prec_Q B)$ are using the unpartitioned objects B and Q as parameters. Due to this independence, the probability bounds can simply be multiplied to derive the joint probability. A formal proof of this lemma is found in Appendix B.3. □

An upper bound can be derived analogously:

$$P_{UB}(RNN_Q(B)) = \prod_{A_i \in I} 1 - P_{LB}(A_i \prec_Q B).$$

[1]This set can be determined using the concept of partial domination [8]. For more details see Algorithm 3.

[2]e.g. using the bounding boxes of the R^*-trees of the objects

In summary, we can now derive, for each uncertain candidate object B a lower and an upper bound of the probability that B is an RNN of Q. However, these bounds may still be rather loose, since we only consider the full uncertainty region of B and Q so far, without any decomposition. In the following section, we will show how to obtain more accurate, still mutually independent probability bounds based on decompositions of B and Q.

4.3.4 Decomposition of Candidate and Query Object

Since the uncertain objects B and Q appear in each random event $A_i \prec_Q B$ ($A_i \in \mathcal{D}$) that has to be evaluated, we cannot split the objects B and Q independently (cf. Section 4.3.2). Intuitively, the reason for this dependency is that any knowledge about the random event $A_i \prec_Q B$ may impose constraints on the position of B and Q. However, Lemma 4 directly yields the following corollary:

COROLLARY 2. *Given partitions $B' \subseteq B$ and $Q' \subseteq Q$, then*

$$P_{LB}(RNN_{Q'}(B')) = \prod_{A_i \in I} 1 - P_{UB}(A_i \prec_{Q'} B').$$

Note that the probability $P_{LB}(RNN_{Q'}(B'))$ is equal to the probability $P_{LB}(RNN_Q(B)|B \in B', Q \in Q')$, where $B \in B', Q \in Q'$ is a constraint to all possible worlds where B is located in partition B' and Q is located in partition Q'. This allows us to individually consider the subset of possible worlds where $B \in B'$ and $Q \in Q'$ and use Lemma 2 to efficiently compute $P_{LB}(RNN_{Q'}(B'))$ and $P_{UB}(RNN_{Q'}(B'))$. This can be performed for each pair $(B', Q') \in \underline{\mathcal{B}} \times \underline{\mathcal{Q}}$, where $\underline{\mathcal{B}}$ and $\underline{\mathcal{Q}}$ denote the decompositions of B and Q, respectively. Now, we can treat pairs of partitions $(B', Q') \in \underline{\mathcal{B}} \times \underline{\mathcal{Q}}$ independently, since all pairs of partitions represent disjoint sets of possible worlds due to the assumption of a disjoint partitioning. Exploiting this independency, we can derive tighter bounds $P_{LB}(RNN_Q(B))$ and $P_{UB}(RNN_Q(B))$ for the probability that B is an RNN of Q by computing a lower and an upper bound of $P(RNN_{Q'}(B'))$ for each $(Q' \in \underline{\mathcal{Q}})$ and each $(B' \in \underline{\mathcal{B}})$ and then computing the weighted sum of these bounds as follows:

$$P_{LB}(RNN_Q(B)) =$$

$$\sum_{B' \in \underline{\mathcal{B}}, Q' \in \underline{\mathcal{Q}}} P_{LB}(RNN_{Q'}(B')) \cdot P(B') \cdot P(Q'). \quad (3)$$

4.4 Verification

For the verification step, we perform, for each remaining candidate B, a probabilistic nearest neighbor query using the algorithm proposed in [12] for probabilistic ranking queries (and setting $k = 1$). This algorithm takes Q, B and $\mathcal{D} \setminus B$ (in particular this set can be reduced, as shown in Appendix D) as input and returns $P(NN_B(Q))$ which is equivalent to $P(RNN_Q(B))$. If this value is above τ, then B is returned as result, otherwise it is discarded. This algorithm avoids enumeration of all (exponentially many) possible worlds by sorting the instances of the influence objects I w.r.t. the distance to B and performing a distance browsing on this sorted list.

4.5 Complexity Analysis

In this section we will analyze the runtime complexity of each part of the proposed PRNN algorithm:

Spatial Pruning: The basic spatial pruning takes each pair of objects $A, B \in \mathcal{D}$ where $A \neq B$ and checks whether $A \prec_Q^{\square} B$. Thus the runtime is $O(|\mathcal{D}|^2)$. In the average case, this step can be accelerated by the use of a spatial index structure to ($O(|\mathcal{D}| \cdot$

$log(|\mathcal{D}|))$, but the worst-case runtime remains $O(|\mathcal{D}|^2)$. The spatial pruning provides us with a set of candidate objects S_{cnd} where each candidate $C_i \in S_{cnd}$ is associated with a set of influence objects S^i_{ifl}.

Probabilistic Pruning: The probabilistic pruning step considers each candidate object C_i separately and partitions the candidate object, the query Q and the influence objects S^i_{ifl}. Assuming a branching factor of the spatial index of b, each object consists of at most b^{depth} partitions, where $depth$ is a parameter chosen before query processing. The pruning relation $A' \prec^{\square}_{Q'} C_i{}'$ is used for each pair of partitions $Q' \in Q, C_i{}' \in C_i$ and each partition A' of objects A in S^i_{ifl}. This leads to a runtime of $O(b^{2 \cdot depth} \cdot |S^i_{ifl}| \cdot b^{depth}) = O(|S^i_{ifl}| \cdot b^{3 \cdot depth})$ for each candidate C_i. In the worst case, where $|S^i_{ifl}| \in O(|\mathcal{D}|)$, $|S_{cnd}| \in O(|\mathcal{D}|)$ and $b^{depth} = m$ this yields a total runtime of $O(|\mathcal{D}|^2 \cdot m^3)$, where m is the number of instances in each object.

Verification: After the probabilistic pruning step, a smaller set of candidates $S'_{cnd} (\subseteq S_{cnd})$ remains. For each candidate $C_i \in S'_{cnd}$ verification is performed. Using the algorithm proposed in [12], this requires to sort all $m \cdot |S^i_{ifl}|$ instances of objects in S^i_{ifl} according to all m instances in C_i. We derive a runtime of $O(|S'_{cnd}| \cdot |S^i_{ifl}| \cdot m^2 \cdot log(|S^i_{ifl}| \cdot m))$. In the worst case, this is in $O(|\mathcal{D}|^2 \cdot log(|\mathcal{D}|)$, assuming m to be constant.

The parameter $depth$**:** It can be observed, that for the case where m is large, the runtime of the probabilistic pruning step may exceed the runtime of the verification step. In our experimental section, we will verify this observation, and show how to choose values for the parameter $depth$ such that this problem is avoided.

5. PROBABILISTIC RkNN QUERIES

In this section we show how our proposed techniques can be extended to probabilistic RkNN queries. An RkNN query is defined as follows: Given a set of (certain) points P, a query object q, and a positive integer k, a reverse nearest neighbor query (RNN_q) returns all $p \in P$ which have q in their k-nearest neighbor set, formally ([3]): $RkNN_q = \{p \in P | dist(p,q) \leq dist(p, p_k)\}$, where p_k is the k-th nearest neighbor of p. In the context of uncertain objects, a $PRkNN^{\tau}_Q$ query returns the set of all objects $U_i \in \mathcal{D}$, for which the probability $P(U_i \in RkNN_Q)$ that U_i is a RkNN of Q is at least τ.

Approximation: Analogous to the $k = 1$ case, we approximate uncertain objects using MBRs and hierarchical partitioning.

Spatial Pruning: In the case where $k > 1$, the random event $A_i \prec_Q B$ must hold for at least k uncertain objects $A_i, 1 \leq i \leq |\mathcal{D}|$ in order to prune candidate B. Therefore, an uncertain candidate object can safely be pruned if there exist at least k objects for which the complete pruning criterion (cf. Lemma 1) holds. The complexity for the spatial pruning step is $O(|\mathcal{D}|^2)$ and independent of k, since in the worst case where a candidate B cannot be pruned, all objects may have to be considered.

Probabilistic Pruning: For probabilistic pruning, the task is to determine for a candidate object B, if its probability to be RkNN of Q is definitely less than τ (at least τ) in order to prune B (return B as a true hit). Analogous to the $k = 1$ case, we can derive the following bounds for the probability $P(A \prec_Q B)$ that $A \in \mathcal{D} \setminus B$ is closer to B than Q: a lower bound $P_{LB}(A \prec_Q B)$ (using Lemma 2) and an upper bound $P_{UB}(A \prec_Q B)$ (using Lemma 3). Given these bounds, we can apply the concept of uncertain generating functions [13] in order to compute for each $0 \leq j < k$ a lower bound $P_{LB}(\#Pruners = j)$ and an upper bound $P_{UB}(\#Pruners = j)$ of the probability of the random event that for exactly j uncertain objects A_i, $A_i \prec_Q B$ is true. A summary of the uncertain generating functions technique is given in Appendix C. Bounds for the probability of the event $RkNN_Q(U_i)$ that U_i is a RkNN of Q can then be derived as follows:

$$P_{LB}(RkNN_Q(U_i)) = \sum_{0 \leq j < k} P_{LB}(\#Pruners = j)$$

$$P_{UB}(RkNN_Q(U_i)) = \sum_{0 \leq j < k} P_{UB}(\#Pruners = j)$$

An uncertain object U_i can be pruned if $P_{UB}(RkNN_Q(U_i)) < \tau$ and returned as a true hit if $P_{LB}(RkNN_Q(U_i)) > \tau$.

As shown in [13], the computational complexity is linear in k, yielding a total of $O(|DB|^2 \times k)$ for the probabilistic pruning.

Verification: The verification step can be performed analogously to the $k = 1$ case using the algorithm proposed in [12], which has been designed for $k \geq 1$. The total complexity of this algorithm is $O(|\mathcal{D}|^2 \cdot log(|\mathcal{D}|) + k \cdot |\mathcal{D}|^2)$.

6. EXPERIMENTS

In the experimental section, we compare our approach which we call HP (hierarchical pruning) with the two state-of-the-art PRNN query algorithms CLWZP [5] and LC [6]. For the implementation of CLWZP and LC we replaced R-trees by R*-trees wherever they were used. For the global R^*-tree we use a page size of 1024 byte. For the local R^*-trees indexing the instances of an uncertain object, we set the page size to a maximal capacity of three entries. The page size was chosen small in order to minimize CPU-cost, which we will see is the main bottleneck of a PRNN query. We tested the three approaches under various parameter and data settings. For each setting, we performed 100 queries and averaged the measures. Since our experiments have been conducted against discrete uncertain datasets, we have adapted the LC algorithm to the discrete case for a fair comparison. The involved parameters and their default values (bold) can be seen in Table 1. For our experiments, we used one real-world dataset and several synthetic datasets to show the effect of changes in dimensionality and size. The datasets are described as follows: We generate synthetic uncertain objects in the $[0, 1]^d$ space by uniformly selecting the expected position of the objects in the space. A rectangle is generated around the expected position with a fixed total sum of side lengths (referred to as *extent*) with a default value of 0.05. By default, the extent is distributed uniformly on the dimensions, so there is a diversity of MBR shapes, some that are nearly cuboid, while others have a very large extent in few dimensions only. The object instances within the MBR are distributed uniformly by default.

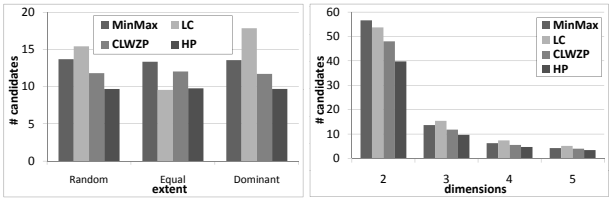
As a real-world dataset, we utilize the International Ice Patrol (IIP) Iceberg Sightings Dataset[3]. This data set contains information about iceberg activity in the North Atlantic in the years 1960 to 2010. It contains the latitude and longitude values of 6216 sighted icebergs. An uncertain object is generated for each iceberg, by generating 100 Normal-distributed instances having a mean value corresponding to the position of its most recent sighting at the date 31.12.2009 and a variance corresponding to the time period between this date and the sighting. To avoid extreme impact of icebergs that have not been seen for decades, any instances outside a square of extent 0.0004 centered at the mean are cut off. The instances of an iceberg are Normal-distributed with a variance based on the time since the its sighting.

The experiments were run on a Windows 7 notebook with an Intel Core i5 processor (2.27 GHz), 6GB RAM. Besides the following experiments, an additional evaluation can be found in Appendix F.

[3]The IIP dataset is available at the National Snow and Ice Data Center (NSIDC) web site (*http://nsidc.org/data/g00807.html*).

parameter	values synthetic	values real
db size	**2000** - 10000	6216
dimensionality	2, **3**, 4, 5	2
# instances	50, **100**, 200, 400	100
τ	0.1 **0.2**, 0.3	**0.2**
$maxdepth$	0, 1, **2**, 3, 4	**2**
MBR$extent$	0.01, 0.02, 0.03, 0.04, **0.05**	N/A

Table 1: Parameters and their default values.



(a) Varying extents. (b) Varying dimensions.

Figure 4: Comparison of different pruning techniques.

6.1 Spatial Pruning

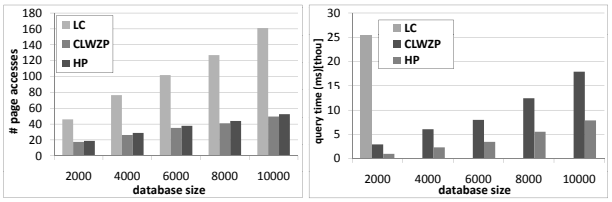
The first set of experiments compares the spatial pruning techniques from the three competing algorithms and the MinDist/MaxDist based pruning (cf. Section 3) called *MinMax*. We generated 2000 uncertain 3-D objects uniformly distributed in the $[0, 1]^3$ space. Each object consists of 100 instances. 100 RNN queries were issued and we compared the number of candidates which were left after the spatial pruning step for the competing techniques. For the experiments shown in Figure 4(a), we tested the influence of different extensions in the dimensions of the objects. Three cases were compared: Random (random extension in each dimension, with fixed maximum), Equal (each dimension had the same extension = hypercube) and Dominant (one dimension has 5 times higher extension than the others). The results show that the spatial pruning technique from our algorithm has the highest pruning power in almost all settings. Only for objects equally extended in each dimension the LC-pruning is competitive. This is due to the reason that in this case, a sphere is a tight approximation for an uncertain object. However, in the other settings the LC-pruning yields the worst performance. We also compared the spatial pruning techniques for data with different dimensionality. Figure 4(b) illustrates that the advantage of HP-pruning is stable over varying dimensionality.

6.2 I/O-Cost

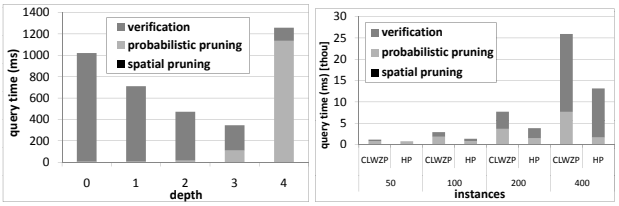
Next, we investigated the number of page accesses of the three algorithms needed on the global R*-tree (the index organizing the uncertain object approximations). The results are shown in Figure 5(a) for synthetic data with different database size. The LC algorithm needs by far the most page accesses which is reasonable, due to the handicaps mentioned in Appendix A.3. The CLWZP algorithm performs even slightly better than the HP algorithm. The reason is mainly founded by the step to get the influence objects for each candidate. CLWZP here performs a search based on all instances of a candidate which produces a tighter bound than only using the approximation of the candidate (as performed in HP). The drawback of this tighter bound is a much higher computational effort, which can be seen in the experiments in the next section. In summary, even for a small page size of 1 kB, the main bottleneck of a probabilistic RNN query are the CPU-cost, not the I/O-cost.

6.3 CPU-Cost

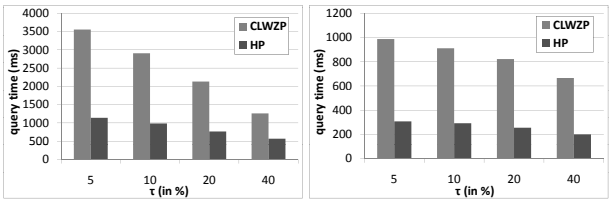
The Parameter $depth$**:** The main tuning parameter of the HP algorithm is $depth$. This parameter offers a tradeoff between the



(a) Comparing page accesses. (b) Comparing cpu time.



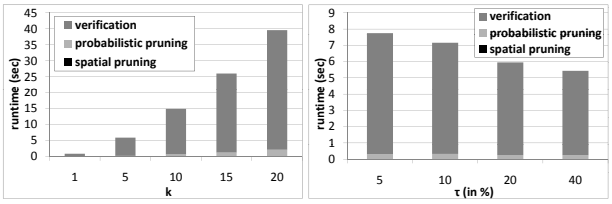
(c) Influence of depth on HP. (d) Effect of sample size.



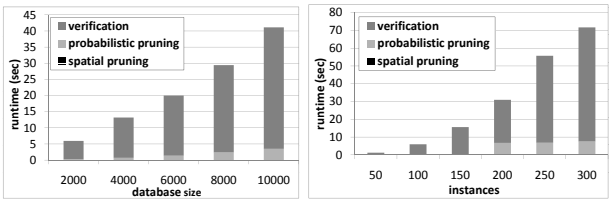
(e) Synthetic Data. (f) Real Data.

Figure 5: Performance of the PRNN Algorithm

computational overhead for the probabilistic pruning and the verification step. This behavior can be seen in Figure 5(c), where by increasing $depth$, it is possible to dramatically reduce the CPU-cost in the verification step and thus the overall costs. It can be observed that for a low value of $depth$, the verification step is the main bottleneck. This is clear, since for a low value of $depth$, the set of partitions of each object X that is used for the probabilistic pruning is very small and contains large partitions. On the one hand, a small number of partitions leads to a very fast probabilistic pruning step, since only a small number of combinations of partitions has to be considered. On the other hand, the pruning power of the probabilistic pruning step is low in this case, due to the coarse approximations. The effect of the $depth$ parameter in this setting is typical: there exists an optimal depth value $depth_{opt}$. Our ex-



(a) Dependency on k. (b) Effect of τ.



(c) Different database sizes. (d) Increasing sample size.

Figure 6: Behaviour of the PRkNN-Algorithm

periments have shown that $depth_{opt}$ correlates to the height H of the R*-Tree. In all of our experiments, choosing $depth = H/2$ has shown to be a good heuristic. Determining better heuristics to find $depth_{opt}$ is part of our future work.

Scalability Experiments: Figure 5(b) shows the effects of the database size on the runtime of the LC, CLWZP and HP algorithms. It can be observed that the LC algorithm has an extremely high CPU-cost. The reason is that the LC algorithm was proposed for continuous uncertain objects and thus requires to consider the set of all possible worlds in the verification step, which is exponential in the number of influence objects. In contrast, both CLWZP and HP scale linear in the database size. Regarding the effect of the number of instances size S, Figure 5(d) shows that the both algorithms CLWZP and HP scale super-linearly. The reason is that both algorithms require to sort instances of a set of influence objects in the verification step, leading to a leading to a runtime of $O(S \cdot log(S))$.

Impact of τ: Figures 5(e) and 5(f) show that the runtime of the HP algorithm decreases for an increasing value of τ for both synthetic and real datasets. The rationale is that a high value of τ reduces the minimal probability $1 - \tau$ required for an uncertain object $A \in \mathcal{D} \setminus B$ to prune B.

PRkNN: We augmented our algorithm to answer PRkNN queries as proposed in Section 5 and evaluated the performance of this algorithm on the synthetic dataset using default parameters (cf. Table 1). In the first experiment (cf. Figure 6(a)), we varied the parameter k. It can be observed that the runtime scales slightly worse than linearly, which can be explained by the usage of uncertain generating functions that show a complexity of $O(k^2)$ ([13]). This is notable, since naive approaches need to consider all $\binom{N}{k}$ possible results. In the remaining experiments (Figures 6(b) to 6(d)) we evaluated the impact of the parameter τ, the database size and the number of instances per object (when setting $k = 5$). Each of these parameters scales equivalently to the $k = 1$ case. Note that in the experiments shown in Figure 6(d) we set $maxdepth = 3$ for more than 150 instances, matching our intuition of setting $maxdepth = H/2$ (cf. Section 6.3).

7. RELATED WORK

Reverse (k)-Nearest Neighbor (R(k)NN) queries on certain data have been studied for quite a while [1, 2, 3]. Current state-of-the-art solutions use a filter-refinement approach to minimize the number of page accesses performed on the index organizing the data. The authors of [3], for example, perform an incremental nearest neighbor query in a best-first search manner where objects are organized in a spatial index and accessed with ascending distance to the query. Each accessed object is then used to prune other objects or index entries in a filter step. Finally, each remaining candidate has to be evaluated by means of a kNN query in a refinement step.

Uncertainty in databases is a relatively new field and has received a lot of attention in the past few years. The main challenges here are data representation [14, 15, 16, 17] and efficient query processing [18, 19]. While the main goal for RkNN queries on certain data is to minimize the I/O-cost, in the context of uncertain data, the CPU-cost also have a dramatic impact on the overall runtime.

Thus, for Probabilistic Reverse Nearest Neighbor (PRNN) queries particularly two challenges arise: minimizing I/O-cost and minimizing CPU-cost. To the best of our knowledge, there are currently two approaches for answering PRNN queries. The approach from Chen et al. [6] which is designed for PRNN queries on uncertain objects represented by continuous probability density functions (PDFs) and the approach from Cheema et al. [5] which works for the discrete case only. Both algorithms are discussed in more detail in Appendix A.

8. CONCLUSIONS

In this paper, we developed a general framework for probabilistic reverse nearest neighbor queries on uncertain data. We showed how two existing approaches and our new algorithm fit into the framework. Through new techniques to improve important parts of the framework, our algorithm is able to outperform the existing approaches under various settings. In addition, we proposed an efficient extension for probabilistic reverse k-nearest neighbor queries. For future work, we want to evaluate techniques to dynamically adapt the parameter $depth$ in the probabilistic pruning step to the distribution of instances within an object.

9. REFERENCES

[1] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," in *Proc. SIGMOD*, 2000.

[2] I. Stanoi, D. Agrawal, and A. E. Abbadi, "Reverse nearest neighbor queries for dynamic databases," in *Proc. DMKD*, 2000.

[3] Y. Tao, D. Papadias, and X. Lian, "Reverse kNN search in arbitrary dimensionality," in *Proc. VLDB*, 2004.

[4] S. Abiteboul, P. C. Kanellakis, and G. Grahne, "On the representation and querying of sets of possible worlds," in *Proc. SIGMOD*, 1987, pp. 34–48.

[5] M. A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei, "Probabilistic reverse nearest neighbor queries on uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 4, pp. 550–564, 2010.

[6] X. Lian and L. Chen, "Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data," *VLDB J.*, vol. 18, no. 3, 2009.

[7] G. Chapman, J. Cleese, T. Gilliam, E. Idle, T. Jones, and M. Palin, "Monty python's the life of brian," 1979.

[8] T. Emrich, H.-P. Kriegel, P. Kröger, M. Renz, and A. Züfle, "Boosting spatial pruning: On optimal pruning of mbrs," in *Proc. SIGMOD*, 2010.

[9] J. Li and A. Deshpande, "Consensus answers for queries over probabilistic databases," in *PODS*, 2009.

[10] A. Guttman, "R-Trees: A dynamic index structure for spatial searching," in *Proc. SIGMOD*, 1984, pp. 47–57.

[11] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: An efficient and robust access method for points and rectangles," in *Proc. SIGMOD*, 1990.

[12] T. Bernecker, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Zuefle, "Scalable probabilistic similarity ranking in uncertain databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1234–1246, 2010.

[13] T. Bernecker, T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Züfle, "A novel probabilistic pruning approach to speed up similarity queries in uncertain databases," in *Proc. ICDE*, 2011 (to appear).

[14] G. Beskales, M. A. Soliman, and I. F. IIyas, "Efficient search for the top-k probable nearest neighbors in uncertain databases," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 326–339, 2008.

[15] R. Cheng, J. Chen, M. Mokbel, and C. Chow, "Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data." in *Proc. ICDE*, 2008.

[16] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. U. Nabar, T. Sugihara, and J. Widom, "Trio: A system for data, uncertainty, and lineage," in *Proc. VLDB*, 2006, pp. 1151–1154.

[17] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, 2007, pp. 15–26.

[18] J. Li, B. Saha, and A. Deshpande, "A unified approach to ranking in probabilistic databases," *PVLDB*, vol. 2, no. 1, pp. 502–513, 2009.

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

[20] J. Chen and R. Cheng, "Efficient evaluation of imprecise location-dependent queries." in *Proc. ICDE*, 2007.

[21] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Querying imprecise data in moving object environments." in *IEEE TKDE*, 2004.

[22] G. Cormode, F. Li, and K. Yi, "Semantics of ranking queries for probabilistic data and expected results," in *Proc. ICDE*, 2009.

[23] X. Lian and L. Chen, "Probabilistic inverse ranking queries over uncertain data," in *Proc. DASFAA*, 2009, pp. 35–50.

[24] S. Singh, C. Mayfield, S. Mittal, S. Prabhakar, S. E. Hambrusch, and R. Shah, "Orion 2.0: native support for uncertain data," in *Proc. SIGMOD*, 2008, pp. 1239–1242.

[25] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[26] J. Li and A. Deshpande, "Ranking continuous probabilistic datasets," in *Proc. VLDB*, 2010 (to appear).

APPENDIX

A. PRNN ALGORITHMS

A.1 LC Algorithm

Approximation: This algorithm is designed for the case where the appearance probability of uncertain objects is represented as a continuous PDF. Though it can easily be adapted to the discrete case. Each uncertain object is approximated by a sphere.

Spatial Pruning: The proposed pruning technique is based on trigonometric functions and can only be applied for spherical objects. Thus, it cannot be directly applied to the index pages (the authors use an R-tree as index structure). To overcome this shortcoming, each (rectangular) page of the index is at runtime approximated by a sphere containing this page.

Probabilistic Pruning: Additionally, a second sphere is computed for each database object in a preprocessing step. This sphere has the same center as the first sphere, but the radius is chosen as the minimal radius covering instances with a cumulated probability of at least $1 - \tau$. The idea of this approach is that if this second sphere can be pruned, then the corresponding object is pruned with a probability of at least $1-\tau$, so it must have a probability less than τ to be an RNN of Q, and thus, it cannot be a PRNN of Q.

Verification: In the verification step, a range query around each candidate U_i is issued. The result contains all objects U_j such that $MinDist(U_j, U_i) < MaxDist(U_i, Q)$, i.e. all objects which affect $P(RNN_Q(U_i))$. Then $P(RNN_Q(U_i))$ is calculated by considering all possible worlds of the involved objects.

A.2 CLWZP Algorithm

Approximation: The CLWZP algorithm uses minimum bounding rectangles for the approximation of the uncertain objects. Additionally, each uncertain object has a local R-tree which organizes its instances.

Spatial Pruning: The pruning is performed using several pruning techniques arranged in series. The first used technique is MinMax. As shown in [8], MinMax is not sufficient, which means that, based on rectangular approximations, MinMax cannot detect valid pruning in all cases. Therefore, a second technique is proposed for special spatial relations of the query object and the pruner. If this technique cannot be applied, a general technique is used which considers all corners of the pruner for prune evaluation (see [5] for details). All proposed techniques (except MinMax) generate a pruning region defined by the pruner and the query. In this region objects can safely be pruned.

Probabilistic Pruning: Probabilistic pruning utilizes the generated pruning regions. Based on these regions, it may happen that only parts of a prunee get pruned. In this case, the prunee is trimmed down and further represented by an MBR containing all instances which could not be pruned (using a computational geometry algorithm). Additionally, the authors propose to partition object Q, to further improve the pruning.

Verification: In the verification phase, a range query is issued for each candidate U_i containing all objects affecting $P(RNN_Q(U_i))$. For each instance u_i of a candidate, the instances of these objects are sorted by the distance to u_i and inserted in a list. Based on these lists it is possible to calculate $P(RNN_Q(U_i))$.

A.3 Discussion

Although the LC algorithm is the only PRNN algorithm so far which can handle uncertain objects represented by a continuous PDF, it has the following drawbacks:

Parameter τ: Since the probabilistic pruning sphere has to be pre-computed using τ, it is not possible to change τ at query time. In a dynamic query environment however, the parameter may be adapted to the user's preferences, which is not possible in this approach.

Spherical Approximation: The main challenge, especially for the higher-dimensional case, is to find a small enclosing sphere of an uncertain object for effective pruning results. Finding the smallest enclosing sphere of an arbitrarily shaped object however has exponential runtime (w.r.t. to the number of vertices of the object), which allows only finding good but not best possible spheres in reasonable computational time. Additionally, as stated in [6], the spatial pruning technique is only conservative but not optimal for dimensions larger than 2. A third problem regarding the spherical approximation is the approximation of pages of the R-tree, which are rectangular by definition. A spherical approximation of an index page will therefore rarely be tight yielding low pruning power.

Verification: The verification step using integration of all remaining objects is based on the used uncertainty model. However, if the objects consist of discrete instances, there are more efficient solutions for the verification step (e.g. [18]). Note that also for the case where objects are represented by continuous PDFs, this step can be performed more efficiently, as we will show later.

The CLWZP algorithm on the other hand has a very complex spatial pruning technique which requires 2^d distance calculations in the worst case (where d is the dimensionality of the data). This makes the approach practically inapplicable for the high-dimensional case. Just as the LC pruning, the CLWZP pruning is conservative which means that there are cases where pruning is not performed although possible (we omit the proof due to space limitations, but will show this by experimental evidence). Regarding the probabilistic pruning of CLWZP, the problem is that trimming requires expensive geometric computation but is used extensively in the algorithm.

B. PROOFS

B.1 Proof of Lemma 1

PROOF. In [8] it is shown that the right hand side of Equation 2 is equivalent to the following statement:

$$\forall a \in A, b \in B, q \in Q : dist(a,b) < dist(q,b)$$

which is true if and only if for each $a \in A, b \in B, q \in Q$ it holds that a is closer to b than q, where A, B, Q are rectangular approximations. By definition of the possible worlds model, the set of combinations $a \in A, b \in B, q \in Q$ corresponds to a superset of all possible worlds. Consequently, it holds that $\forall a \in A, b \in B, q \in Q : \delta(a,b,q) = 1$. Using Equation 1 we obtain the triple-sum

$$P(A \prec_Q B) = \sum_{a_i \in A} \sum_{b_j \in B} \sum_{q_k \in Q} 1 \cdot P(a_i) \cdot P(b_j) \cdot P(q_k)$$

As we can see, the above triple-sum is equal to the sum of the probabilities of *all* possible worlds which is equal to one[4]. Consequently, we obtain $P(A \prec_Q B) = 1$. □

B.2 Proof of Lemma 2

PROOF. The probability of a combination (A', B', Q') can be computed by $P(A') \cdot P(B') \cdot P(Q')$ due to the assumption of mutually independent objects. These probabilities can be aggregated

[4]Here we assume no existential uncertainty on the uncertain objects. If we have existential uncertainty, $P(A \prec_Q B) = P(Q) \cdot P(A) \cdot P(B)$.

due to the assumption of disjoint subregions, which implies that any two different combinations of subregions ($A' \in \underline{\mathcal{A}}$, $B' \in \underline{\mathcal{B}}$, $Q' \in \underline{\mathcal{Q}}$) and ($A'' \in \underline{\mathcal{A}}$, $B'' \in \underline{\mathcal{B}}$, $Q'' \in \underline{\mathcal{Q}}$, $A' \neq A'' \wedge B' \neq B'' \wedge Q' \neq Q''$) must represent disjoint sets of possible worlds. By definition it holds that, if $\delta(A', B', Q') = 1$, then A prunes B in all possible worlds defined by combinations of instances ($a_i \in A', b_j \in B', q_k \in Q'$). But not all possible worlds where A prunes B are covered by these combinations and, thus, do not contribute to $P_{LB}(A \prec_Q B)$. Consequently, $P_{LB}(A \prec_Q B)$ lower bounds $P(A \prec_Q B)$. □

B.3 Proof of Lemma 4

PROOF. The event that B is an RNN of Q is (by definition) equal to the event that no database object A_i in I prunes B. Due to Corollary 3 (see below), the event $A_i \prec_Q B$ is independent of the computation of $P_{LB}(A_j \prec_Q B)$ for $i \neq j$. Therefore, the bounds $P_{LB}(A_i \prec_Q B)$ are also independent of $P_{LB}(A_j \prec_Q B)$. This independence allows to derive a lower bound of the joint probability that all objects $A_1, ..., A_{|I|}$ prune B by simply taking the product of the bounds $\prod_{i=1}^{|I|} P_{LB}(A_i \prec_Q B)$. The same applies for the joint probability of the complementary events (i.e. the probability of the events that the A_i's do not prune B). Since we only have bounds of $P(A_i \prec_Q B)$, we need to use the upper bounds $P_{UB}(A_i \prec_Q B)$ in order to minimize the complementary probability $1 - P_{UB}(A_i \prec_Q B)$ to derive a lower bound of the event that no A_i prunes B.

Analogously, we can derive an upper bound of the probability that none of the $A_1, ..., A_k$ prunes B by multiplying the counter probabilities $\prod_{i=1}^{k} 1 - P_{LB}(A_i \prec_Q B)$. □

COROLLARY 3. *Let $A_1, ..., A_{|I|}$ be uncertain objects with disjoint object decompositions $\mathcal{A}_1, ..., \mathcal{A}_{|I|}$, respectively. Also, let B and Q be uncertain objects without any decomposition, i.e. $\underline{\mathcal{B}} = \{B\}$ and $\underline{\mathcal{Q}} = \{Q\}$. The random event $A_i \prec_Q B$ is independent of the result of the computation of the probability bounds $P_{LB}(A_i \prec_Q B)$ and $P_{UB}(A_i \prec_Q B)$.*

PROOF. Consider the random variable $A_i \prec_Q B$ conditioned on the event $P_{LB}(A_j \prec_Q B) = p$.

$$P(A_i \prec_Q B | P_{LB}(A_j \prec_Q B) = p)$$

Substituting the equation in Lemma 2, this yields

$$P(A_i \prec_Q B | \sum_{A_j' \in \underline{\mathcal{A}}, B' \in \underline{\mathcal{B}}, Q' \in \underline{\mathcal{Q}}} P(A') \cdot P(B') \cdot P(Q') \cdot \delta(A_j', B', Q') = p)$$

Since $\underline{\mathcal{B}} = \{B\}$ and $\underline{\mathcal{Q}} = \{Q\}$, this becomes equal to

$$P(A_i \prec_Q B | \sum_{A_j' \in \underline{\mathcal{A}}} P(A') \cdot P(B) \cdot P(Q) \cdot \delta(A_j', B, Q) = p)$$

Exploiting that the probability that B (Q) falls into partition B (Q) is one, this implies

$$P(A_i \prec_Q B | \sum_{A_j' \in \underline{\mathcal{A}}} P(A') \cdot \delta(A_j', B, Q) = p)$$

Since $A_i \prec_Q B$ is independent of the position of A_j (and thus of the events that A is located in A'), and since $\delta(A_j', B, Q)$ is not a random event, we conclude that the above is equal to

$$P(A_i \prec_Q B)$$

Finally, the equation

$$P(A_i \prec_Q B | P_{UB}(A_j \prec_Q B) = p) = P(A_i \prec_Q B)$$

can be shown analogously. □

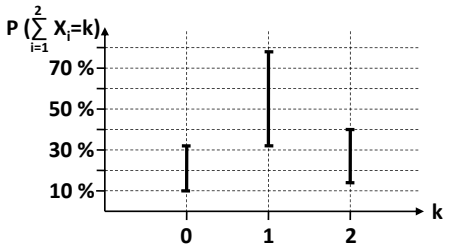


Figure 7: Approximated PDF of $\sum_{i=1}^{2} X_i$.

C. UNCERTAIN GENERATING FUNCTIONS

Given a set of N mutually independent but not necessarily identically distributed Bernoulli $\{0, 1\}$ random variables X_i, $1 \leq i \leq N$. Let $P_{LB}(X_i)$ ($P_{UB}(X_i)$) be a lower (upper) bound approximation of the probability $P(X_i = 1)$. We consider the random variable $\sum_{i=1}^{N} X_i$ and make the following observation: $X_i = 1$ with a probability of at least $P_{LB}(X_i)$, and $X_i = 0$ with a probability of at least $1 - P_{UB}(X_i)$. Based on this observation, we consider the following uncertain generating function (UGF):

$$\mathcal{F}^N = \prod_{i \in 1,...,N} [P_{LB}(X_i) \cdot x + (1 - P_{UB}(X_i)) \cdot y + (P_{UB}(X_i) - P_{LB}(X_i))] = \sum_{i,j \geq 0} c_{i,j} x^i y^j.$$

The coefficient $c_{i,j}$ has the following meaning: With a probability of $c_{i,j}$, B is definitely dominated at least i times, and possibly dominated another 0 to j times. Therefore, the minimum probability that $\sum_{i=1}^{N} X_i = k$ is $c_{k,0}$, since that is the probability that exactly k random variables X_i are 1. The maximum probability that $\sum_{i=1}^{N} X_i = k$ is $\sum_{i \leq k, i+j \geq k} c_{i,j}$, i.e. the total probability of all possible combinations in which $\sum_{i=1}^{N} X_i = k$, may hold. Therefore, we obtain an approximated PDF of $\sum_{i=1}^{N} X_i$. In the approximated PDF of $\sum_{i=1}^{N} X_i$, each probability $\sum_{i=1}^{N} X_i = k$ is given by a conservative and a progressive approximation.

EXAMPLE 1. *Let $P_{LB}(X_1) = 20\%$, $P_{UB}(X_1) = 50\%$, $P_{LB}(X_2) = 60\%$ and $P_{UB}(X_2) = 80\%$. The generating function for the random variable $\sum_{i=1}^{2} X_i$ is the following:*

$$\mathcal{F}^2 = (0.2x + 0.5y + 0.3)(0.6x + 0.2y + 0.2)$$
$$= 0.12x^2 + 0.34x + 0.1 + 0.22xy + 0.16y + 0.06y^2$$

This implies that, with a probability of at least 12%, $\sum_{i=1}^{2} X_i = 2$. In addition, with a probability of 22% plus 6%, it may hold that $\sum_{i=1}^{2} X_i = 2$, so that we obtain a probability bound of $12\% - 40\%$ for the random event $\sum_{i=1}^{2} X_i = 2$. Analogously, $\sum_{i=1}^{2} X_i = 1$ with a probability of $34\% - 78\%$ and $\sum_{i=1}^{2} X_i = 0$ with a probability of $10\% - 32\%$. The approximated PDF of $\sum_{i=1}^{2} X_i$ is depicted in Figure 7.

In order to apply uncertain generating functions to our problem of determining the distribution of the number of objects that prune a given candidate object B with respect to a query object Q, we only need to substitute the random variables X_i by the random variables $A_i \prec_Q B$. This is feasible, since our derived approximations of the probability of the random variables $A_i \prec_Q B, 1 \leq i \leq N$ are independent of the other random variables $A_j \prec_Q B$, $1 \leq j \leq N, j \neq i$ due to Lemma 4.

D. IMPLEMENTATION

In this section, we describe the implementation of our PRNN algorithm.

D.1 Overview

Algorithm 1 combines the main modules for PRNN processing. The input requires an uncertain query object Q, an R-tree based index structure organizing the uncertain objects from the database I_{DB}, a probability threshold τ and a parameter $depth$ controlling the depth of our probabilistic pruning computation. The spatialPruning method fills the sets S_{cnd} with potential result objects and S_{prn} with objects (entries) which can certainly be excluded using the spatial pruning technique proposed in Section 4.2. For each remaining object $B \in S_{cnd}$, the getInfluenceObjects method returns a set (S_{ifl}) of all objects from the two sets (S_{cnd} and S_{prn}) which could influence $P(RNN_Q(B))$. With these objects, probabilistic pruning according to Section 4.3.2 is performed. Depending on the result, the candidate is either discarded, added to the result set or verified. In the latter case, computation following [18] and [5] is performed to calculate the exact $P(RNN_Q(B))$. If this probability is above τ, the candidate can be confirmed as a result. In the following, we explain the individual modules in detail.

Algorithm 1 PRNN query processing

Require: Q, I_{DB}, τ, $depth$
1: $S_{cnd} = \emptyset$, $S_{prn} = \emptyset$
2: spatialPruning(Q, I_{DB}, S_{cnd}, S_{prn})
3:
4: $S_{res} = \emptyset$
5: **for** each $B \in S_{cnd}$ **do**
6: S_{ifl} = getInfluenceObjects(Q, B, $S_{cnd} \setminus \{B\}$, S_{prn})
7: i := probabilisticPruning(Q, B, S_{ifl}, τ, $depth$)
8: **if** i=1 **then**
9: *//B cannot be RNN of Q*
10: **else if** i=-1 **then**
11: *//B is RNN of Q*
12: $S_{res} = S_{res} \cup \{B\}$
13: **else**
14: *//B has to be verified*
15: **if** verify(Q, B, S_{ifl}, τ) **then**
16: $S_{res} = S_{res} \cup \{B\}$
17: **end if**
18: **end if**
19: **end for**
20: **return** S_{res}

D.2 Spatial Pruning

The spatialPruning method (cf. Algorithm 2) performs a best-first search using a heap H prioritized by $minDist(Q, e)$, where e is an entry of the index I_{DB}. The heap is implemented such that the set of contained objects can be accessed without destroying the heap structure. For each de-heaped entry e, the predicate $e_2 \prec_Q e$ checks if this entry is pruned by another object or entry e_2 (contained in H, S_{prn} or S_{cnd}) according to Q, using the pruning technique as described in Section 4.2. If it can be pruned, it is inserted in the S_{prn} set. Otherwise, if e contains a data object, it is added to the candidate set S_{cnd} and if e is a directory entry, its children are inserted into the heap.

D.3 Getting the Influence Objects

For each candidate B, it is important for the next steps to find the objects which influence $P(RNN_Q(B))$. This is done by Algorithm 3, which exploits Corollary 1 in order to determine those objects that cannot possibly prune B. Therefore, each data entry e for which $Q \prec_e B$ does not hold is inserted into the S_{ifl} set.

Algorithm 2 spatialPruning

Require: Q, I_{DB}, S_{cnd}, S_{prn}
1: init min-heap H with root entry of I_{DB}
2: **while** H is not empty **do**
3: de-heap an entry e from H
4: **if** $\exists e_2 \in H \cup S_{prn} \cup S_{cnd} : e_2 \prec_Q e$ **then**
5: $S_{prn} = S_{prn} \cup \{e\}$
6: **else if** e is directory entry **then**
7: **for** each child ch in e **do**
8: insert ch in H
9: **end for**
10: **else if** e is data entry **then**
11: $S_{cnd} = S_{cnd} \cup \{e\}$
12: **end if**
13: **end while**

Algorithm 3 getInfluenceObjects

Require: Q, B, S_{cnd}, S_{prn}
1: $S_{ifl} = \emptyset$
2: **for** each $e \in S_{prn} \cup S_{cnd}$ **do**
3: **if** $\neg(Q \prec_e B)$ **then**
4: **if** e is directory entry **then**
5: $S_{prn} = S_{prn} \setminus \{e\}$
6: **for** each child ch in e **do**
7: $S_{prn} = S_{prn} \cup \{ch\}$
8: **end for**
9: **else if** e is data entry **then**
10: $S_{ifl} = S_{ifl} \cup \{e\}$
11: **end if**
12: **end if**
13: **end for**
14: **return** S_{ifl}

D.4 Probabilistic Pruning

The goal of the probabilistic pruning is to estimate $P(RNN_Q(B))$ for a candidate B in a best possible way. If we can detect early that $P(RNN_Q(B)) > \tau$ or $P(RNN_Q(B)) < \tau$, further computation can be saved. The parameter $depth$ is used to control how deep the hierarchical index structures (organizing the instances of each uncertain object) are resolved for more accurate pruning. Thus, the parameter offers to define a trade-off between accuracy and computational efficiency in the probabilistic pruning step. In each iteration the involved objects (Q, B and all $I \in S_{ifl}$) are partitioned, which means we consider the partitioning at the i^{th} level of each local R*-tree. According to Section 4.3.2, each combination of ($Q' \in \underline{\mathcal{Q}}, B' \in \underline{\mathcal{B}}$) must be treated independently. Thus, for each of these combinations, we first obtain bounds $P_{LB}(I \prec'_Q B')$ and $P_{UB}(I \prec'_Q B')$ of the probability that each $I \in S_{ifl}$ prunes B' w.r.t. Q', using Lemmas 2 and 3. Using Corollary 4, we multiply the complement of these bounds to acquire the lower (upper) bound probability $P_{LB}(RNN_{Q'}(B'))$ ($P_{UB}(RNN_{Q'}(B'))$) that no object $I \in S_{ifl}$ prunes B' according to Q'. Since all combinations ($Q' \in \underline{\mathcal{Q}}, B' \in \underline{\mathcal{B}}$) are independent, the results can be summed up (weighting with the possible world probability of ($Q' \in \underline{\mathcal{Q}}, B' \in \underline{\mathcal{B}}$)), to obtain the global probability bounds $P_{LB}(RNN_Q(B))$ and $P_{UB}(RNN_Q(B))$ according to Equation 3. This method returns -1 if the candidate B is PRNN of Q (with τ as threshold), and 1 if B can be pruned. If $depth$ is not set to the maximum height of the R*-trees, it is possible that no decision can be made. In this case the method returns 0.

E. CONTINUOUS DISTRIBUTIONS

In this section, we show how our approach can be extended to continuously distributed uncertainty models. Again, we assume

Algorithm 4 probabilisticPruning

Require: Q, B, S_{ifl}, τ, $depth$
1: **for** $1 \ldots depth$ **do**
2: split(Q)
3: split(B)
4: $\forall I \in S_{ifl}$ split(I)
5: $P_{LB}(RNN_Q(B)) = 0, P_{UB}(RNN_Q(B)) = 0$
6: **for all** $Q' \in Q$ and $B' \in B$ **do**
7: $P_{LB}(RNN_{Q'}(B')) = 1, P_{UB}(RNN_{Q'}(B')) = 1$
8: **for** each $I \in S_{ifl}$ **do**
9: $P_{LB}(I \prec_{Q'} B') = 0, P_{UB}(I \prec_{Q'} B') = 1$
10: **for** each $I' \in I$ **do**
11: **if** $(I' \prec_{Q'} B')$ **then**
12: $P_{LB}(I \prec_{Q'} B') = P_{LB}(I \prec_{Q'} B') + P(I')$
13: **else if** $(Q' \prec_{I'} B')$ **then**
14: $P_{UB}(I \prec_{Q'} B') = P_{UB}(I \prec_{Q'} B') - P(I')$
15: **end if**
16: **end for**
17: $P_{UB}(RNN_{Q'}(B')) = P_{UB}(RNN_{Q'}(B')) \cdot (1.0 - P_{LB}(I \prec_{Q'} B'))$
18: $P_{LB}(RNN_{Q'}(B')) = P_{LB}(RNN_{Q'}(B')) \cdot (1.0 - P_{UB}(I \prec_{Q'} B'))$
19: **end for**
20: $P_{LB}(RNN_Q(B)) = P_{LB}(RNN_Q(B)+P(Q')) \cdot P(B') \cdot P_{LB}(RNN_{Q'}(B'))$
21: $P_{UB}(RNN_Q(B)) = P_{UB}(RNN_Q(B)+P(Q')) \cdot P(B') \cdot P_{UB}(RNN_{Q'}(B'))$
22: **end for**
23: **if** $P_{LB}(RNN_Q(B)) > \tau$ **then**
24: **return** -1
25: **else if** $P_{UB}(RNN_Q(B)) < \tau$ **then**
26: **return** 1
27: **end if**
28: **end for**
29: **return** 0

that the database $\mathcal{D}$ consists of multi-attribute objects $o_1, ..., o_N$ that may have uncertain attribute values. An uncertain attribute is defined as follows:

DEFINITION 3 (CONTINUOUS PROBABILISTIC ATTRIBUTE). *A continuous probabilistic attribute* attr *of an object X is a random variable drawn from a probability distribution with density function f_i^{attr}.*

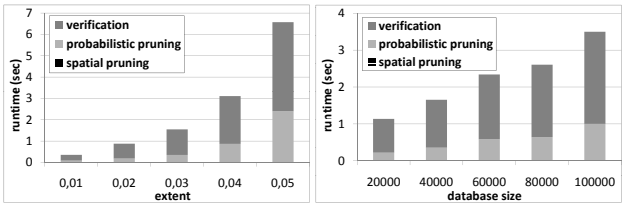
An uncertain object X has at least one uncertain attribute value. The function $f_X(x)$ denotes the multi-dimensional probability density function (PDF) of o_i that combines all density functions for all probabilistic attributes $attr$ of X.

Approximation: Following the convention of continuous uncertain databases [14, 20, 15, 21, 22, 23, 24], we assume that each uncertain object X is (minimally) bounded by a rectangular *uncertainty region* $X^\square$ such that $\forall x \notin X^\square : f_X(x) = 0$ and

$$\int_{X^\square} f_X(x)dx \leq 1.$$

If f_i is an unbounded PDF, e.g., a Gaussian PDF, we truncate PDF tails with negligible probabilities and normalize the resulting PDF. This procedure is also used in related work [20, 15, 14]. Specifically, [14] shows that, for a reasonably low truncation threshold, the impact on the accuracy of probabilistic ranking queries is very low.

Spatial Pruning: Since our spatial pruning approach (cf. Section 4.2) is based on rectangular approximations only, it can be applied on continuously distributed objects without any adaptations.



(a) Varying extents. (b) Varying dbsize.

Figure 8: Additional experiments for the PRkNN algorithm

Probabilistic Pruning: Our probabilistic pruning approach (cf. Section 4.3) applies disjoint and complete partitioning schemes $\underline{\mathcal{X}}$ ($X \in \mathcal{D} \cup Q$) to conservatively and progressively approximate the probability $P(RNN_Q(B))$ that an object B is an RNN of Q. This technique is also applicable for partitions $X' \in \underline{\mathcal{X}}$ for which the probabilities $P(X')$ are known. To derive a complete and disjoint partitioning scheme on continuous uncertain objects, we propose to apply a kd-tree [25] having $X^\square$ in its root. In each level of the tree, each node is split with respect to its median in dimension d. The split-dimension d is rotated for each level of the tree. The advantage of this partitioning scheme is that the probability of a node on level i (here level 0 denotes the root level) of the tree, has a total probability of $1/2^i$. Due to the nature of continuous PDFs, this kd-tree has an infinite height. Therefore, we propose to restrict the height of the kd-tree (i.e. the maximum number of splits). The maximum number of splits is denoted by $depth$.

Verification: To compute the exact probability $P(A \prec_Q B)$ that an object A prunes B for a PRNN query with query object Q, we require to compute the following integral:

$$\int_{a \in A^\square} \int_{b \in B^\square} \int_{q \in Q^\square} f_A(a) \cdot f_B(b) \cdot f_Q(q) \cdot \delta(a,b,q) da\, db\, dq$$

where $\delta(a,b,q)$ is the indicator function defined in Section 4.3 that returns 1 iff $dist(a,b) < dist(b,q)$. This computation requires expensive numeric integrations, since in general the integral of the PDF f_X of an uncertain object may not be representable as a closed-form expression and the integral of $\delta(a,b,q)$ does not have a closed-form expression. The computation of $RNN_Q(B)$ is even more expensive, since to compute $RNN_Q(B)$, the PDFs of all database objects may need to be considered to avoid dependencies. Therefore, we propose to avoid verification by using a large $depth$ parameter, so that the derived probability bounds $P_{LB}(RNN_Q(B))$ and $P_{UB}(RNN_Q(B))$ become very tight and with a high probability, no more candidates remain after the probabilistic pruning step. For the remaining candidates, the best we can do is an efficient approximation ([26]). Therefore, we propose the following strategies:

- Our approach uses the derived probability bounds of candidates to bound the error. In many applications, this bound may be sufficient to the user and verification may be avoided.
- We can adapt the techniques as proposed by [26] to approximate the object PDFs using cubic splines that have a closed-form solution, and approximate the rank of Q w.r.t. B.

F. ADDITIONAL EXPERIMENTS

Figure 8 illustrates additional experiments on synthetic datasets focusing on the extent of the uncertain objects and much larger databases. The smaller the extent of the uncertain objects the more efficient queries can be performed (cf. Figure 8(a)). For the experiment shown in Figure 8(b), we lowered the average extent of the objects to 0.01. For this extent, the query times are still very good, even for very large datasets (100000 objects * 100 instances = 10 million data points).