

Probabilistic Ranking in Uncertain Vector Spaces

Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Andreas Zuefle
{bernecker, kriegel, renz, zuefle}@dbs.ifi.lmu.de

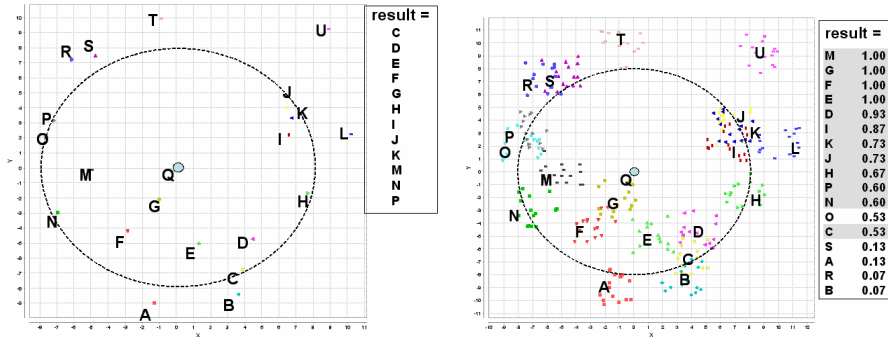
Institute for Informatics, Ludwig-Maximilians-Universität München, Germany

Abstract. In many application domains, e.g. sensor databases, traffic management or recognition systems, objects have to be compared based on positionally and existentially uncertain data. Feature databases with uncertain data require special methods for effective similarity search. In this paper, we propose a probabilistic similarity ranking algorithm which computes the results dynamically based on the complete information given by inexact object representations. Hence, this can be performed in an effective and efficient way. We assume that the objects are given by a set of points in a vector space with confidence values following the discrete uncertainty model. Based on this representation, we introduce a probabilistic ranking algorithm that is able to reduce significantly the computational complexity of the computation of the probability that an object is at a certain ranking position. In a detailed experimental evaluation, we demonstrate the benefits of this approach compared to several competitors. The experiments show that, in addition to the gain of efficiency, we can achieve convenient query results of high quality.

1 Introduction

Similarity ranking is one of the most important query types in feature databases. A similarity ranking query iteratively reports objects in descending order of their similarity to a given query object. The iterative computation of the answers is very suitable for retrieving the results the user could have in mind. This is a big advantage of ranking queries against the most prominent similarity queries, the distance-range (ε -range) and the k -nearest neighbor query, in particular if the user does not know how to specify the query parameters ε and k .

Many modern applications have to cope with uncertain or imprecise data. Example applications are location determination and proximity detection of moving objects, similarity search and pattern matching in sensor databases or personal identification and recognition systems based on video images or scanned image data. The importance of this topic in the context of database systems is demonstrated by the increasing interest of the database research community in this subject matter. Several approaches coping with uncertain objects have been proposed [4, 5, 12, 2]. All these approaches use continuous probability density functions (pdfs) for the description of the spatial uncertainty while the approaches proposed in [7, 8] use discrete representations of uncertain objects. The approach proposed in [7] supports probabilistic distance range queries on uncertain objects. In [8] efficient methods for probabilistic nearest-neighbor queries are proposed. However, in fact only one-nearest neighbor queries are supported.



(a) Query on objects with aggregated uncertainty

(b) Query on objects with probabilistic uncertainty

Fig. 1. Distance range query on objects with different uncertainty representations.

Similarity search in conjunction with multimedia data like images, music, or data from personal identification systems like face snapshots or fingerprints commonly involves distance computations within the feature space. If exact features cannot be generated from uncertain objects, we have to cope with *positionally* uncertain vectors in the feature space (i.e. objects are represented by ambiguous feature vectors). Basically, there exist two forms of representations of positionally uncertain data: Uncertain positions represented by a probability density function (pdf) or uncertain positions drawn by alternatives. In this paper we concentrate on uncertain objects represented by a set of alternative positions, each associated with a confidence value that indicates the degree of matching the exact object. This type of representation is motivated by the fact that we often have only discrete but ambiguous object information as usually returned by common sensor devices, e.g. discrete snapshots of continuously moving objects.

A probabilistic ranking on uncertain objects computes for each object $o \in \mathcal{D}$ the probability that o is the k -th nearest neighbor ($1 \leq k \leq |\mathcal{D}|$) of a given query object q . In the context of probabilistic ranking queries we propose diverse forms of ranking outputs which differ in the order the objects are reported to the user. Furthermore, we suggest diverse forms in which the results are reported (i.e. which kind of information is assigned to each result).

The simplest solution to perform queries on uncertain objects is to represent the objects by an exact feature vector, e.g. the mean vector, and perform query processing in a traditional way. The advantage of this straightforward solution is that established query and indexing techniques can be applied. However, this solution is accompanied by information loss, since the similarity between uncertain objects is obviously more meaningful when taking the whole information of the object uncertainty into account. An example of the latter case is depicted in Figure 1(a), where a set of uncertain objects $A \dots U$ represented by their mean values is depicted. The results of a distance range query with query object Q are shown in the upper right box. There are critical objects

like P , that is included in the result, and O , which is not included, though they are very close to each other and have quite similar uncertainty regions as depicted in Figure 1(b)¹. Here, the complete uncertainty information of the objects is taken into account. The gray shaded fields indicate those objects which are included in the non-probabilistic result (cf. Figure 1(a)). The results show that objects O and P have quite similar probabilities ($P(O) = 50\%$, $P(P) = 60\%$) for belonging to the result. Additionally, we can see that objects E , F , G and M are certain results.

2 Related Work

Several approaches for indexing uncertain vector objects have been proposed. They mainly differ in the type of uncertainty supported by the index and in the type of supported similarity query. In [3], the Gauss-tree is introduced which is an index structure for managing large amounts of Gaussian distribution functions. The proposed system aims at efficiently answering so-called identification queries. Additionally, [3] proposed *probabilistic identification queries* which are based on a Bayesian setting (i.e., given a query pdf, retrieve those pdfs in the database that correspond to the query pdf with the highest probability).

The authors of [4, 5, 12, 2] deal with an uncertainty model for positionally uncertain objects and propose queries which are specified by intervals in the query space. In this setting, a query retrieves uncertain objects w.r.t. the likelihood that the uncertain object is indeed placed in the given query interval. The authors of [2] adapt the Gauss-tree proposed in [3] to a positionally uncertainty model and discuss *probabilistic ranking queries*. Here, the *probabilistic ranking query* has another meaning than the queries proposed in this paper. In [2], probabilistic ranking queries retrieve those k objects which have the highest probability of being located inside a given query area.

In [12] an index structure called U-Tree is proposed which organizes pdfs using linear approximations. Recently, [10] introduced a method and a corresponding index structure modeling pdfs using piecewise-linear approximations. This new approach also employs linear functions as the U-Tree but is more exact in its approximation.

All the approaches mentioned above use continuous probability density functions (pdfs) for the description of the spatial uncertainty. Most of them only support specific types of pdfs, e.g. uniform distribution within an interval or Gaussian distributions.

The approaches proposed in [7, 8] use discrete representations of positionally uncertain objects. Instead of continuous probability density functions they use sampled object positions reflecting the positionally object uncertainty. Based on this concept they proposed efficient similarity search approaches that allow to approximate uncertain objects represented by pdfs of arbitrary structure. The main advantage of this approach is that sampled positions in space can efficiently be indexed using traditional spatial access methods thus allowing to reduce the computational complexity of complex query types. The approach proposed in [7] supports probabilistic distance range queries on uncertain objects. The advantage of probabilistic distance range queries is that the result probability for an uncertain object does not depend on the other uncertain

¹ Here the object uncertainties are indicated by a set of alternative positions, i.e. each uncertain object consists of a set of alternative positions.

objects in the database. The approach proposed in [8] enables an efficient computation of probabilistic nearest-neighbor queries. However, only one-nearest neighbor queries are supported. The main challenge for k -NN queries is that the neighborhood probability of objects depends on the other objects in the database.

Recently, Soliman et al. presented in [11] a top- k query processing algorithm for uncertain data in relational databases. The uncertain objects are represented by multiple tuples where to each tuple a confidence value is assigned indicating the likelihood that the tuple is a representant of the corresponding object. The authors propose two different query methods, the uncertain top- k query (*U-Topk*) and the uncertain k ranks query (*U-kRanks*). The *U-Topk* query reports tuples with the maximum aggregated probability of being top- k for a given score function while the *U-kRanks* query reports for each ranking position one tuple which is a clear winner of the corresponding ranking position. For both query types efficient query processing algorithms are presented. An improved method is given for both query types by Yi et al. in [13]. Though the *U-kRanks* query problem defined in [11, 13] is quite related to our problem it is based on the x -relation model and, thus, differs from our problem definition. Their approaches refer to the occurrence of single tuples in a possible world instead of objects composed by a set of mutually exclusive vector points.

In this paper, we propose efficient solutions for probabilistic ranking queries. Thereby, the results are iteratively reported in ascending order of the ranking parameter k . Similar to the object uncertainty model used in [7, 8], our approach assumes that the uncertain objects are represented by a set of points in a vector space. This allows us to use standard spatial access methods like the R*-tree [9] for the efficient organization of the uncertain objects. Furthermore standard similarity search paradigms can be exploited to support probabilistic ranking in an efficient way.

3 Problem Definition

In this section, we formally introduce the problem of probabilistic ranking queries on uncertain objects. We first start with the definition of (positionally) uncertain objects.

3.1 Positionally Uncertain Objects

Objects of a d -dimensional vector space \mathbb{R}^d are called *positionally uncertain*, if they do not have a unique position in \mathbb{R}^d , but have multiple positions associated with a probability value. Thereby, the probability value assigned to a position $p \in \mathbb{R}^d$ of an object o denotes the likelihood that o is located at the position p in the feature space. A formal definition is given in the following:

Definition 1 (Uncertain Object Representation). *Let \mathcal{D} be a database of objects located in a d -dimensional feature space \mathbb{R}^d . Corresponding to the discrete uncertainty model, an uncertain object o is modelled by a finite set of alternative positions in a d -dimensional vector space each associated with a confidence value, i.e. $o = \{(x, p) : x \in \mathbb{R}^d, p \in [0, 1], p \text{ is the probability that } x \text{ is the position of } o\}$. The confidence value p indicates the likelihood that the vector position matches the corresponding position of object o . The condition $\sum_{(x,p) \in o} p = 1$ holds.*

3.2 Distance Computation for Uncertain Objects

Positionally uncertain objects involve uncertain distances between them. Like the uncertain position, the distance between two uncertain objects (or between two objects where at least one of them is an uncertain object) can be described by a probability density function (pdf) that reflects the probability for each possible distance value. However for uncertain objects with discrete uncertainty representations we need another form of distance.

Definition 2 (Uncertain Distance). Let $dist : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_0^+$ be an L_p -norm based similarity distance function, and let $o_i \in \mathcal{D}$ and $o_j \in \mathcal{D}$ be two uncertain objects, where o_i and o_j are assumed to be independent of each other. Then an uncertain distance in the discrete uncertainty model is a collection $d_{uncertain}(o_i, o_j) = \{(d, p) \in \mathbb{R}_0^+ : \forall (x, p_x) \in o_i, \forall (y, p_y) \in o_j : d = dist(x, y), p = p_x \cdot p_y\}$. Here, the condition $\sum_{(x,p) \in d_{uncertain}(o_i, o_j)} p = 1$ holds.

The probability, that returns the likelihood that the uncertain distance $d_{uncertain}(o_i, o_j)$ between two uncertain objects o_i and o_j is smaller than a given range $\varepsilon \in \mathbb{R}_0^+$ can be estimated by:

$$P(d_{uncertain}(o_i, o_j) \leq \varepsilon) = \sum_{\substack{(x, p) \in d_{uncertain}(o_i, o_j) \\ d \leq \varepsilon}} p.$$

Since distance computations between uncertain objects are very expensive, we need computationally inexpensive distance approximations to reduce the candidate set in a filter step. For this reason, we introduce distance approximations that lower and upper bound the uncertain distance between two uncertain objects.

Definition 3 (Minimal and Maximal Object Distance). Let $o_i = \{o_{i,1}, o_{i,2}, \dots, o_{i,M}\}$ and $o_j = \{o_{j,1}, o_{j,2}, \dots, o_{j,M'}\}$ be two uncertain objects. Then the distance $d_{min}(o_i, o_j) = \min_{s=1..M, s'=1..M'} \{dist(o_{i,s}, o_{j,s'})\}$ is called minimal distance between the objects o_i and o_j . Analogously, the distance $d_{max}(o_i, o_j) = \max_{s=1..M, s'=1..M'} \{dist(o_{i,s}, o_{j,s'})\}$ is called maximal distance between the objects o_i and o_j .

3.3 Probabilistic Ranking on Uncertain Objects

The output of probabilistic queries is usually in form of a set of result objects, each associated with a probability value indicating the likelihood that the object fulfills the query predicate. However, in contrast to ε -range queries and k -nn queries, ranking queries do not have such an unique query predicate, since the query predicate changes with each ranking position. In case of a ranking queries, to each result object a set of probability values is assigned, one for each ranking position. We call this form of ranking output *probabilistic ranking*.

Definition 4 (Probabilistic Ranking). Let q be an uncertain query object and \mathcal{D} be a database containing $N = |\mathcal{D}|$ uncertain objects. An uncertain ranking is a function

$prob_ranked_q : (\mathcal{D} \times \{1, \dots, N\}) \rightarrow [0..1]$ that reports for a database object $o \in \mathcal{D}$ and a ranking position $k \in \{1, \dots, N\}$ the probability which reflects the likelihood that o is at the k^{th} ranking position according to the uncertain distance $d_{uncertain}(o, q)$ between o and the query object q in ascending order.

The probabilistic ranking includes the following information, a probability value for each object and for each ranking position. Not all of this information might be relevant for the user and it could be difficult for the user to extract the relevant information. Commonly, a small part of the probabilistic ranking information should be sufficient and more easy to read and, thus, more convenient for most applications. Furthermore, due to the variance of the ranking positions of the objects, there does not exist a unique order in which the results are reported. For this reason, we define different types of probabilistic ranking queries which differ in the order the results are reported and in the form their confidence values are aggregated.

In the following definitions, we assume an uncertain query object q and a database \mathcal{D} containing $N = |\mathcal{D}|$ uncertain objects are given. Furthermore, we assume that $prob_ranked_q$ is a probabilistic ranking over \mathcal{D} according to q . The following variants of query definitions can be easily motivated by the fact that the user could be overstrained with ambiguous ranking results. They specify how the results of a probabilistic ranking can be aggregated and reported in a more comfortable form which is more easy to read. In particular, for each ranking position only one object is reported, i.e. for each ranking position k , the object which *best fits* the given position k is reported.

Probabilistic Ranking Query Based on Maximal Confidence (PRQ_MC) The first query definition reports the objects in such a way that the k^{th} reported object has the highest confidence to be at the given ranking position k . This query definition is quite similar to the *U-kRanks* query defined in [11, 13].

Definition 5. A probabilistic ranking query based on maximal confidence (PRQ_MC) incrementally retrieves for the next ranking position $i \in \mathcal{I}_N$ a result tuple of the form $(o, prob_ranked_q(o, i))$, where $o \in \mathcal{D}$ has not been reported at previous ranking iterations (i.e. at ranking positions $j < i$) and $\forall p \in \mathcal{D}$ which have not been reported at previous ranking iterations, the following statement holds:

$$prob_ranked_q(o, i) \geq prob_ranked_q(p, i).$$

Note that this type of query only considers the occurrence confidence of a certain ranking position for an object. The confidences of prior ranking positions of an object are ignored in the case they are exceeded by another object. However, the confidences of prior ranking positions might also be relevant for the final setting of the ranking position of an object. This assumption is taken into account with the next query definition.

Probabilistic Ranking Query Based on Maximal Aggregated Confidence (PRQ_MAC)

The next query definition *PRQ_MAC* takes aggregated confidence values of ranking positions into account. Contrary to the previous definition, this query assigns each object o a unique ranking position k by aggregating over the confidences of all prior ranking positions $i < k$ according to o .

Definition 6. A probabilistic ranking query based on maximal aggregated confidence (PRQ_MAC) incrementally retrieves for the next ranking position $i \in \mathcal{I}_N$ a result tuple of the form $(o, \sum_{j=1..i} \text{prob_ranked}_q(o, j))$, where $o \in \mathcal{D}$ has not been reported at previous ranking iterations (i.e. at ranking positions $j < i$) and $\forall p \in \mathcal{D}$ which have not been reported at previous ranking iterations, the following statement holds:

$$\sum_{j=1..i} \text{prob_ranked}_q(o, j) \geq \sum_{j=1..i} \text{prob_ranked}_q(p, j).$$

Both query types defined above specify the ranking position of each object o by comparing the ranking position confidence of o with that of the other objects. The next query specification takes for the assignment of a ranking position to an object o only the ranking confidences of o into account.

Probabilistic Ranking Query Based on Expected k -Matching (PRQ_EkM) This query assigns to each object its expected ranking position without taking the confidences of the other objects into account.

Definition 7. A probabilistic ranking query based on expected k -matching (PRQ_EkM) incrementally retrieves for the next ranking position $i \in \mathcal{I}_N$ a result tuple of the form $(o, \text{prob_ranked}_q(o, i))$, where $o \in \mathcal{D}$ has not been reported at previous ranking iterations (i.e. at ranking positions $j < i$) and o has the i^{th} highest expected ranking position

$$\mu(o) = \sum_{j=1..N} j \cdot \text{prob_ranked}_q(o, j).$$

In other words, the objects are reported in ascending order of their expected ranking position.

4 Probabilistic Ranking Algorithm

The computation of the probabilistic ranking is very expensive and is the main bottleneck of the probabilistic ranking queries proposed in the previous section. In this section, we first introduce in Section 4.1 the data model used to compute the probabilistic ranking and then, in Section 4.2, we show how the computational cost of the probabilistic ranking on uncertain objects can be drastically reduced. We assume that each object is represented by M alternative vector positions which we call sample points or simply samples in the remainder. Furthermore, we assume that the object samples are stored in a spatial index structure like the R^* -tree [9], in order to organize the uncertain objects such that proximity queries can be efficiently processed.

Up to now, we have assumed that the database objects are uncertain. If we assume that the query object is an uncertain object as well, we have to keep the dependencies between the alternative object representations given by the sample representations in mind. For this reason, we propose to solve the probabilistic ranking problem for each representant $o_{q,j}$ of the query object $o_q = \{o_{q,1}, o_{q,2}, \dots, o_{q,M}\}$ separately. Let us note

that this computation can be done in parallel and, thus, can be efficiently supported by distributed systems.

In the following, we concentrate on the computation of the probabilistic ranking query according to one sample point $q_j \in \mathbb{R}^d$ of the query object q . The computation is done for each query sample point separately and, in a postprocessing step, the results can be easily merged to obtain the final result which is shown in Section 4.3.

4.1 Iterative Probability Computation

Initially, an iterative computation of the nearest neighbors of q_j w.r.t. the sample points of all objects $o \in \mathcal{D}$ (sample point ranking $rank_s(q_j)$) is started using the ranking algorithm proposed in [6]. Then, we iteratively pick object samples from the sample point ranking $rank_s(q_j)$ according to the query sample point q_j . For each sample point $o_{i,s}$ ($1 \leq s \leq M$) returned from $rank_s(q_j)$, we immediately compute the probability that $o_{i,s}$ is the k^{th} nearest neighbor of q_j for all k ($1 \leq k \leq i$). Thereby, all other samples $o_{i,t}$ ($t \neq s$) of object o_i have to be ignored due to the sample dependency within an object as mentioned above.

For the probability computation we need two auxiliary data structures, the *sample table* (ST), required to compute the probabilities by incorporating the other objects $o_l \in \mathcal{D}$ ($o_l \neq o_i$), and the *probability table* (PT) used to maintain the intermediate results w.r.t. $o_{i,s}$ and which finally contains the overall results of the probabilistic ranking. In the following, both data structures ST and PT are introduced in detail.

Sample Table (ST) We maintain a table ST called *sample table* that stores for each accessed object separately the portion of samples already returned from $rank_s(q_j)$. Additionally, we need for each accessed object the portion of samples that has not been accessed so far. Entries of ST according to object o_i are defined as follows:

$$ST[i][1] = \frac{\# \text{ samples of } o_i \text{ already returned from } rank_s(q_j)}{M (\hat{=} \# \text{ samples of object } o_i)},$$

$ST[i][0]$ can be directly computed by $ST[i][0] = 1 - ST[i][1]$, such that in fact we only need to maintain entries of the form $ST[i][1]$.

Probability Table (PT) Additionally to the sample table, we maintain a table PT called *probability table* that stores for each object o_i and each $k \in \mathbb{N}$ ($1 \leq k \leq N$) the actual probability that o_i is the k^{th} -nearest neighbor of the query sample point q_s . The entries of PT according to the s^{th} sample point of object o_i are defined as follows:

$$PT[k][i][s] = P((k-1) \text{ objects } o \in \mathcal{D}, (o \neq o_i), \text{ are closer to } q_j \text{ than the sample point } o_{i,s}).$$

We assume that object o_i is the i^{th} object for which $rank_s(q_j)$ has reported at least one sample point. The same assumption is made for the sample points of an uncertain object (i.e., sample point $o_{i,s}$ is the s^{th} -closest sample point of object o_i according to q_j). These assumptions hold for the remainder of this paper.

Now, we show how to compute an entry $PT[k][i][s]$ of the probability table using the information stored in the sample table ST . Let ST be a sample table of size N (i.e. ST stores the information corresponding to all N objects of the database \mathcal{D}). Let $\sigma_k(i) \subseteq \{o \in \mathcal{D} | o \neq o_i\}$ denote the set, called k -set of o_i , containing exactly $(k-1)$ objects. If we assume $k < N$, obviously $\binom{N}{k}$ different k -set permutations $\sigma_k(i)$ exist. For the computation of $PT[k][i][s]$, we have to consider the set S_k of all possible k -set permutations according to o_i . The probability that exactly $(k-1)$ objects are closer to the query-sample point q_j than the sample point $o_{i,s}$, can be computed as follows:

$$PT[k][i][s] = \sum_{\sigma_k(i) \in S_k} \prod_{\substack{l=1..N \\ l \neq i}} \begin{cases} ST[l][1], & \text{if } o_l \in \sigma_k(i) \\ ST[l][0], & \text{if } o_l \notin \sigma_k(i) \end{cases}$$

Let us assume that we actually process the sample point $o_{i,s}$. Since the object samples are processed in ascending order according to their distance to q_j , the sample table entry $ST[l][1]$ reflects the probability, that object o_l is closer to q_j than the sample point $o_{i,s}$. On the other hand, $ST[l][0]$ reflects the probability that $o_{i,s}$ is closer to q_j than o_l .

In the following, we show how the entries of the probability table can be computed by fetching iteratively the sample points from $rank_s(q_j)$. Thereby, we assume that all entries of the probability table are initially set to zero. Then the iterative ranking process $rank_s(q_j)$ which reports one sample point of an uncertain object in each iteration, is started. Each reported sample point $o_{i,s}$ is used to compute for all k ($1 \leq k \leq N$) the probability value that corresponds to the table entry $PT[k][i][s]$. After filling the $(i-s)$ -column of the probability table, we proceed with the next sample point fetched from $rank_s(q_j)$ in the same way as we did with $o_{i,s}$. This procedure is repeated until all sample points are fetched from $rank_s(q_j)$.

4.2 Accelerated Probability Computation

The computation of the probability table can be very costly in space and time. One reason is the size of the table that grows drastically with the number of objects and the number of samples for each object. Another problem is the very expensive computation of the probability table entries $PT[k][i][s]$. In the following, we propose some methods that reach a considerable reduction of the overall query cost.

Table Pruning Obviously, we do not need to maintain separately the result according to each sample point of an object. Instead of maintaining a table entry for each sample point of an object, we have to compute the average over the sample probabilities according to an object and a ranking position. This can be done on the fly by simply summing up the iteratively computed sample probabilities. An additional reduction of the table (i.e., a reduction to those parts of the table that should be available at once) can be achieved by maintaining only those parts of the table that are required for further computations and skip the rest. First, we have to maintain a table column only for those objects from which at least one sample point has been reported from $rank_s(q_j)$,

whereas we can skip those from which we already fetched all sample points. In the same way we can reduce the sample table in order to reduce the cost required to compute the probability table entries. Second, we can skip each probability table row that corresponds to a ranking position which is not within a certain ranking range. This range is given by the minimal and maximal ranking position of the uncertain objects for which we currently have to maintain a column of the probability table. The following lemmas utilize the bounds for uncertain distances that are introduced in Definition 3.

Lemma 1 (Minimal Ranking Position). *Let $o_i \in \mathcal{D}$ be an uncertain object and q_j be the query sample point. Furthermore, let at least $n \in \mathcal{I}_N$ objects have a maximal distance that is smaller or equal to the minimal distance of o_i , i.e. $|\{o_l : o_l \in \mathcal{D}, d_{max}(o_l, q_j) \leq d_{min}(o_i, q_j)\}| \geq n$. Then, the ranking position of object o_i must be larger than n .*

In the same way, we can upper bound the ranking position of an uncertain object.

Lemma 2 (Maximal Ranking Position). *Let $o_i \in \mathcal{D}$ be an uncertain object and q_j be the query sample point. Furthermore, let at most $n' \in \mathcal{I}_N$ objects have a minimal distance that is smaller or equal to the maximal distance of o_i , i.e. $|\{o_l : o_l \in \mathcal{D}, d_{min}(o_l, q_j) \leq d_{max}(o_i, q_j)\}| = n'$. Then, the ranking position of object o_i must be lower than or equal to n' .*

As mentioned above, the computation of the object probabilities according to ranking position i only requires to consider those objects whose minimal and maximal ranking position cover the ranking position i . This holds for those objects having sample points within as well as outside of the actual range of the actual ranking distance $r - dist$. Usually, in practice this is the case for only a small set of objects depending on their spatial density and specificity of their uncertainty.

Bisection-Based Algorithm The computational cost can be significantly reduced if we utilize the bisection-based algorithm as proposed in [1]. The bisection-based algorithm uses divide-and-conquer which computes for a query object q and a database object o the probability that exactly k other objects are closer to q than the object o . The main idea is to recursively perform a binary split of the set of relevant objects, i.e. objects which have to be taken into account for the probability computation. Afterwards, the corresponding results can be efficiently merged into the final result. Here, we leave out details due to limited space. Note that this approach, although this approach accelerates the computation cost of the $PT[k][i][s]$ significantly, the asymptotical cost is still exponential in the ranking range. This approach is mentioned here because it is an important competitor to the dynamic-programming-based approach presented next (cf. Section 5).

Dynamic-Programming-Based Algorithm In the following, we introduce our new approach which is able to efficiently compute the $PT[k][i][s]$ and whose runtime is $O(|\mathcal{D}|^3)$. The key idea of our approach is based on the following property. Given a sample q and a set of j objects $\mathcal{S} = \{o_1, o_2, \dots, o_j\}$ for which the probability $P(o_i, q)$ that $o_i \in \mathcal{S}$ is ranked higher than q is known. Now, we want to compute the probability $P_{k, \mathcal{S}, q}$ that exactly k $o_i \in \mathcal{S}$ are ranked higher than q .

Lemma 3. *If we assume that object o_j is ranked higher than q , then $P_{k,S,q}$ is equal to the probability that exactly $k-1$ objects of $S \setminus \{p_j\}$ are ranked higher than q . Otherwise, $P_{k,S,q}$ is equal to the probability that exactly k objects of $S \setminus \{p_j\}$ are ranked higher than q .*

The above lemma leads to the following recursion that allows to compute $P_{k,S,q}$ by means of the paradigm of dynamic programming:

$$P_{k,S,q} = P_{k-1,S \setminus \{p_j\},q} \cdot p_j + P_{k,S \setminus \{p_j\},q} \cdot (1 - p_j), \text{ where } P_{0,\emptyset} = 1.$$

Let us note that the above dynamic programming scheme was originally proposed in the context of Top- k queries in the x-relational model [13]. Here, we can exploit this scheme to compute the probability that an uncertain object $o \in \mathcal{D}$ is assigned to a certain ranking position.

4.3 Building Final Query Results

Up to now, we have assumed that the query consists of one query sample point. In the following, we show how we support queries where the query object is also uncertain, i.e. consists of several query sample points. Let us assume that the query object q consists of M query sample points. Then we start for each sample point $q_j \in q$ separately a probabilistic ranking query as described above. The results are finally merged simply by computing for each object the average over all corresponding probabilities returned from the M queries, i.e.

$$prob_ranked_q(i)(o) = \frac{\sum_{j=1..M} prob_ranked_{q_j}(i)(o)}{M}.$$

5 Experimental Evaluation

In this section, we examine the effectiveness and efficiency of our proposed probabilistic similarity ranking approaches. Since the computation is highly CPU bounded, we measured the efficiency by the overall runtime cost required to compute an entire ranking averaged over 10 queries.

5.1 Datasets

The following experiments are based on artificial and real-world datasets. The artificial datasets which are used for the efficiency experiments contain 10 to 1000 10-dimensional uncertain objects that are situated by a Gaussian distribution in the data space. Each object consists of $M = 10$ alternative positions that are distributed around the mean positions of the objects with a variance of 10% of the data space if not stated otherwise. Figure 2 depicts the distribution of uncertain objects when varying the variance of the positions of the uncertain objects, i.e. the degree of uncertainty. A growing variance leads to an increase of the overlap between the object samples. For the evaluation of the effectiveness of our methods we used two real-world datasets: O_3 and

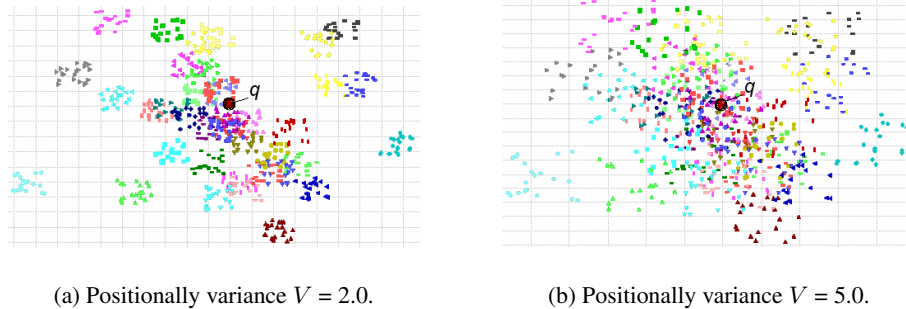


Fig. 2. Uncertain object distribution (object variance = 10.0) in 60×60 space for different degrees of positionally uncertainty. (number of objects $N = 40$, number of samples $S = 20$)

datasets	<i>PRQ_MC</i>	<i>PRQ_MAC</i>	<i>PRQ_EkM</i>	<i>MP</i>
O_3	0.51	0.65	0.53	0.63
NSP_h	0.36	0.43	0.29	0.35
NSP_{frq}	0.62	0.70	0.41	0.60

Fig. 3. Avg. precision for probabilistic ranking queries on different real-world datasets.

NSP. The O_3 dataset is an environmental dataset consisting of 30 uncertain time series, each composing a set of measurements of O_3 concentration in the air measured within one month. Thereby, each measurement features a daily O_3 concentration curve. The dataset covers measurements from the year 2000 to 2004 and is classified according to the months in a year. The *NSP* dataset is a chronobiologic dataset describing the cell activity of *Neurospora*² within sequences of day cycles. This dataset is used to investigate endogenous rhythms. It can be classified according to two parameters among others: day cycle and type of mold. For our experiments we utilized two subsets of the *NSP* data: NSP_h and NSP_{frq} . NSP_h is classified according to the day cycle length. It consists of 36 objects that created three classes of day cycle (16, 18 and 20 hours). The NSP_{frq} dataset consists of 48 objects and is classified according to the type of the mold ($frq1$, $frq7$ and $frq+$).

5.2 Effectiveness

In the first experiments, we evaluate the quality of the different probabilistic ranking queries (*PRQ_MC*, *PRQ_MAC*, *PRQ_EkM*) proposed in Section 3.3. In order to make a fair evaluation, we compare them with the results of a non-probabilistic ranking (*MP*) which ranks the objects based on the distance between their mean positions. For these

² *Neurospora* is the name of a fungal genus containing several distinct species. For further information see *The Neurospora Home Page*: <http://www.fgsc.net/Neurospora/neurospora.html>.

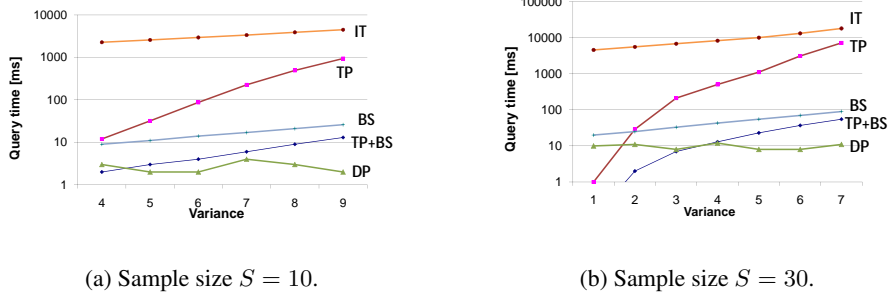


Fig. 4. Query processing cost w.r.t. the degree of object uncertainty.

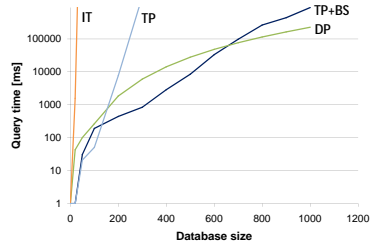
experiments, we used the three real-world datasets O_3 , NSP_h and NSP_{freq} , each consisting of uncertain objects which are classified as described above. The quality of the proposed approaches can be directly compared in the table depicted in Figure 3 which shows the average precision over all recall values based on a k -nn classification for each probabilistic ranking query approach and each dataset. In all experiments, the PRQ_{MAC} approach outperforms the other approaches including the non-probabilistic ranking approach. Interestingly, the approach PRQ_{MC} which has a quite similar definition as the U - $kRanks$ query proposed in [11, 13] does not work very well and shows similar quality as the non-probabilistic ranking approach. The approach PRQ_{EKM} loses clearly and is even significantly below the non-probabilistic ranking approach. This observation points out that the postprocessing step, i.e. the way in which the results of the probabilistic rankings are post-processed, indeed affects the result.

5.3 Efficiency

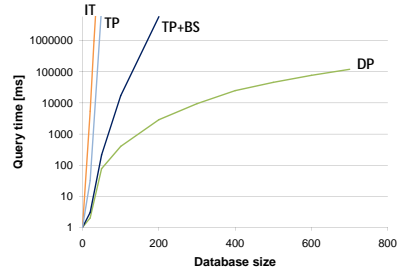
In the next experiment, we evaluate the performance of our probabilistic ranking acceleration strategies proposed in Section 4.2 w.r.t. query processing time. We experimentally evaluate the performance of our algorithms by comparing the different proposed strategies against the straightforward solution where the computation of the query without any additional strategy. A summary of the competing methods is given below (cf. Section 4.2):

- IT** Iterative fetching of the sample points from the sample point ranking $rank_s(q_j)$ and computation of the probability table PT entries without any acceleration strategy.
- TP** Table pruning strategy where we used the reduced table space.
- BS** Bisection-based computation of the probability permutations.
- DP** Dynamic-Programming-based computation of the probability permutations.

Influence of Degree of Uncertainty In the first experiment, we compare all strategies (including the straightforward solution) for probabilistic rankings on the artificial



(a) Variance = 0.5



(b) Variance = 5.0

Fig. 5. Comparison of the scalability of all strategies (uniform uncertainty distribution).

datasets with different grade of uncertainty (variance). The evaluation of the query processing time of our approaches is illustrated in Figure 4. In particular, the differences between the used computation strategies are depicted for two different sample sizes $S = 10$ and $S = 30$. Here, we used a database size of 20 uncertain objects of a 10-dimensional feature space. Obviously, the **DP** shows the best performance. Using only the recursive computation **BS**, the query processing time is quite high even for a low variance value. However, the query time increases only slightly when further increasing the variance. On the other hand, using only the table pruning strategy **TP** leads to a significant increase in computation time, in particular for high variances. The computation time of **TP** is much smaller for low variances compared to **BS**. The **DP** approach is not affected by an increasing variance. To sum up, using a combination of the **TP** and **BS** strategies results in a quite good performance, but it is outperformed by the **DP** approach due to its polynomial computational complexity.

Scalability Next, we evaluate the scalability based on the artificial datasets of different size. Here, we also considered different combinations of strategies. The experimental results are depicted in Figure 5. Obviously the simple approach **IT** produces such overwhelming cost compared to the other strategies that experiments for a database size above 30 objects are not applicable. It can clearly be observed that the combination **TP+BS** significantly outperforms just **TP**. The scalability over a larger range of database sizes can be seen on the right hand side of Figure 5. The basic **TP** approach is already not applicable for very small databases, even for a low degree of object uncertainty. It is interesting to see that for very small database sizes and low degree of uncertainty the **TP+BS** outperforms **DP**. Let us note that we would achieve quite less cost for the query processing if we limit the ranking output to a $k \ll N$. Anyway, a complete ranking of the database is usually not required. In contrast to the other competitors the **DP** scales well even for large databases.

6 Conclusions

In this paper, we proposed an approach that efficiently computes probabilistic ranking queries on uncertain objects represented by sets of sample points. In particular, we proposed methods that are able to break down the high computational complexity required to compute for an object o the probability, that o has the ranking position k ($1 \leq k \leq N$) according to the distance to a query object q . We theoretically and experimentally showed that our approach is able to speed-up the query by factors of several orders of magnitude. In the future we plan to apply probabilistic ranking queries to improve data mining applications.

References

1. T. Bernecker, H.-P. Kriegel, and M. Renz. Proud: Probabilistic ranking in uncertain databases. In *In Proc. 20th Int. Conf. on Scientific and Statistical Database Management (SSDBM'08), Hong Kong, China, July 9-11*, pages 558–565, 2008.
2. C. Böhm, A. Pryakhin, and M. Schubert. "Probabilistic Ranking Queries on Gaussians". In *Proc. of the 18th Int. Conf. on Scientific and Statistical Database Management (SSDBM'06)*, pages 169–178, 2006.
3. C. Böhm, A. Pryakhin, and M. Schubert. "The Gauss-Tree: Efficient Object Identification of Probabilistic Feature Vectors". In *Proc. 22nd Int. Conf. on Data Engineering (ICDE'06), Atlanta, GA, US*, page 9, 2006.
4. R. Cheng, D. Kalashnikov, and S. Prabhakar. "Evaluating Probabilistic Queries over Imprecise Data". In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA*, pages 551–562, 2003.
5. R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. Vitter. "Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data". In *Proc. 30th Int. Conf. on Very Large Databases (VLDB'04), Toronto, Canada*, pages 876–887, 2004.
6. G. Hjaltason and H. Samet. "Ranking in Spatial Databases". In *Proc. 4th Int. Symposium on Large Spatial Databases, SSD'95, Portland, USA*, volume 951, pages 83–95, 1995.
7. H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz. "Probabilistic Similarity Join on Uncertain Data". In *Proc. 11th Int. Conf. on Database Systems for Advanced Applications (DASFAA'06), Singapore, Singapore*, pp. 295-309, 2006, (Best paper).
8. H.-P. Kriegel, P. Kunath, and M. Renz. "Probabilistic Nearest-Neighbor Query on Uncertain Objects". In *Proc. 12th Int. Conf. on Database Systems for Advanced Applications (DASFAA'07), Bangkok, Thailand*, pp. 337-348, 2007.
9. H.-P. Kriegel, B. Seeger, R. Schneider, and N. Beckmann. "The R*-tree: An Efficient Access Method for Geographic Information System". In *Proc. Int. Conf. on Geographic Information Systems, Ottawa, Canada*, 1990.
10. V. Ljosa and A. K. Singh. APLA: Indexing arbitrary probability distributions. In *Proc. of the 23rd Int. Conf. on Data Engineering (ICDE 2007)*, 2007.
11. M. Soliman, I. Ilyas, and K. Chen-Chuan Chang. "Top-k Query Processing in Uncertain Databases". In *Proc. 23rd Int. Conf. on Data Engineering (ICDE'07), Istanbul, Turkey*, pages 896–905, 2007.
12. Y. Tao, R. Cheng, X. Xiao, W. Ngai, B. Kao, and S. Prabhakar. "Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions". In *Proc. 31th Int. Conf. on Very Large Data Bases (VLDB'05), Trondheim, Norway*, pages 922–933, 2005.
13. K. Yi, F. Li, G. Kollios, and D. Srivastava. "Efficient Processing of Top-k Queries in Uncertain Databases". In *Proc. 24th Int. Conf. on Data Engineering (ICDE'08), Cancún, México*, 2008.