Probabilistic Verifiers: Evaluating Constrained Nearest-Neighbor Queries over Uncertain Data

Reynold Cheng[†], Jinchuan Chen[†], Mohamed Mokbel[‡], Chi-Yin Chow[‡]

[†]*Deptartment of Computing, Hong Kong Polytechnic University*
Hung Hom, Kowloon, Hong Kong
{csckcheng,csjcchen}@comp.polyu.edu.hk

[‡]*Department of Computer Science and Engineering,University of Minnesota-Twin Cities*
200 Union Stree SE, Minneapolis, MN 55455
{mokbel,cchow}@cs.umn.edu

***Abstract*— In applications like location-based services, sensor monitoring and biological databases, the values of the database items are inherently uncertain in nature. An important query for uncertain objects is the Probabilistic Nearest-Neighbor Query (PNN), which computes the probability of each object for being the nearest neighbor of a query point. Evaluating this query is computationally expensive, since it needs to consider the relationship among uncertain objects, and requires the use of numerical integration or Monte-Carlo methods. Sometimes, a query user may not be concerned about the exact probability values. For example, he may only need answers that have sufficiently high confidence. We thus propose the Constrained Nearest-Neighbor Query (C-PNN), which returns the IDs of objects whose probabilities are higher than some threshold, with a given error bound in the answers. The C-PNN can be answered efficiently with *probabilistic verifiers*. These are methods that derive the lower and upper bounds of answer probabilities, so that an object can be quickly decided on whether it should be included in the answer. We have developed three probabilistic verifiers, which can be used on uncertain data with arbitrary probability density functions. Extensive experiments were performed to examine the effectiveness of these approaches.**

I. Introduction

Data uncertainty is an inherent property in various emerging applications. Consider a habitat monitoring system where data like temperature, humidity, and wind speed are acquired from sensors. Due to the imperfection of the sensing devices, the data obtained are often contaminated with noises [1]. In the Global-Positioning System (GPS), the location values collected have some measurement error [2], [3]. In biometric databases, the attribute values of the feature vectors stored are also not exact [4]. These errors should be captured and treated carefully, in order to provide high-quality query answers.

Sometimes, uncertainty is introduced by the system. In Location-Based Services (LBS), it is expensive to monitor every change in location. Instead, the "dead-reckoning" approach is used, where each mobile device only sends an update to the system when its value has changed significantly. The location is modeled in the database as a range of possible values [2], [5]. Recently, the idea of injecting location uncertainty to a user's location in an LBS has been proposed [6], [7], in order to protect a user's location privacy.

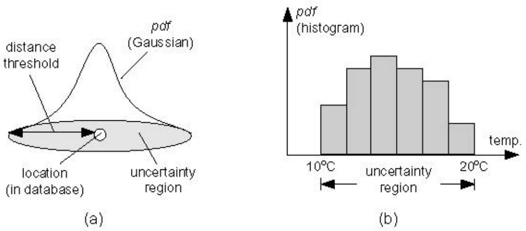


Fig. 1. Location and sensor uncertainty.

A well-studied uncertainty model is to assume that the actual data value is located within a closed region, called the *uncertainty region*. In this region, a non-zero probability density function (*pdf*) of the value is defined, where the integration of pdf inside the region is equal to one. In an LBS where the dead-reckoning approach is used, a normalized Gaussian distribution is used to model the measurement error of a location stored in a database [2], [3] (Figure 1(a)). Gaussian distributions are also used to model values of a feature vector in biometric databases [4]. Figure 1(b) shows the histogram of temperature values in a geographical area observed in a week. The pdf, represented as a histogram, is an arbitrary distribution between 10^oC and 20^oC. In this paper, we focus on uncertain objects in the one-dimensional space (i.e., a pdf defined inside a closed interval).

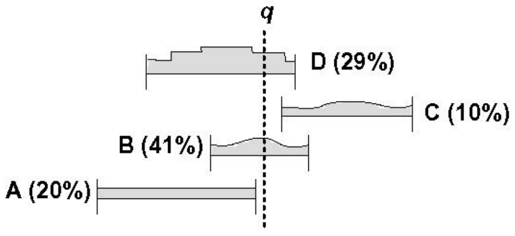


Fig. 2. Probabilistic NN Query (PNN).

An important query for uncertain objects is the *Probabilistic Nearest Neighbor* Query (PNN in short) [5]. This query returns

978-1-4244-1837-4/08/$25.00 © 2008 IEEE

the non-zero probability (called *qualification probability*) of each object for being the nearest neighbor of a given point q. The qualification probability augmented with each object allows us to place confidence onto the answers. Figure 2 illustrates an example of PNN on four uncertain objects (A, B, C and D). The query point q and the qualification probability of each object are also shown. A PNN could be used in a scientific application, where sensors are deployed to collect the temperature values in a natural habitat. For data analysis and clustering purposes, a PNN can be executed to find out the district(s) whose temperature values is (are) the closest to a given centroid. Another example is to find the IDs of sensor(s) that yield the minimum or maximum wind-speed from a given set of sensors [5], [1]. A minimum (maximum) query is essentially a special case of PNN, since it can be characterized as a PNN by setting q to a value of $-\infty$ (∞).

Although PNN is useful, evaluating it is not an easy task. In particular, since the exact value of a data item is not known, one needs to consider the item's possible values in its uncertainty region. Moreover, an object's qualification probability depends not just on its own value, but also on the relative values of other objects. If the uncertainty regions of the objects overlap, then their pdfs must be considered in order to derive their corresponding probabilities. In Figure 2, for instance, evaluating A's qualification probability (20%) requires us to consider the pdfs of the other three objects, since each of them has some chance of overtaking A as the nearest neighbor of q.

To our best knowledge, there are two major techniques for computing qualification probabilities. The first method is to derive the pdf and cumulative density function (*cdf*) of each object's distance from q. The probability of an object is then computed by integrating over a function of distance pdfs and cdfs [5], [8], [1]. A recent solution proposes to use the Monte-Carlo method, where the pdf of each object is represented as a set of points. The qualification probability is evaluated by considering the portion of points that could be the nearest neighbor [9]. The cost of these solutions can be quite high, since they either require numerical integration over some aggregate functions of arbitrary pdfs, or the handling of samples which are acquired from the object. Moreover, the accuracy of the answer probabilities depends on the precision of the integration or number of samples used. It is worth notice that an indexing solution for pruning objects with zero qualification probabilities have been proposed in [8]. This technique can remove a large fraction of objects from consideration. However, the evaluation time for the remaining objects, as shown in their experiments, still consumes a lot of CPU resources.

A. Solution Overview

Although calculating qualification probabilities is expensive, a query user may not always be interested in the precise probability values. A user may only require answers with confidence that are higher than some fixed value. In Figure 2, for instance, if an answer with probability higher than 30% is required, then object B (41%) would be the only answer. If a user can tolerate with some approximation in the result (e.g., he allows an object's actual probability to be 2% less than the 30% threshold), then object D (29%) can be another answer. Here the *threshold* (30%) and *tolerance* (2%) are requirements or *constraints* imposed on a PNN. We denote this variant of PNN as the **Constrained Probabilistic Nearest-Neighbor Query** (or **C-PNN** in short). A C-PNN allows the user to control the desired confidence and quality in the query. The answers returned, which consists of less information than PNN, may also be easier to understand. In Figure 2, the C-PNN only includes objects (B,D) in its result, as opposed to the PNN which returns the probabilities of all the four objects.

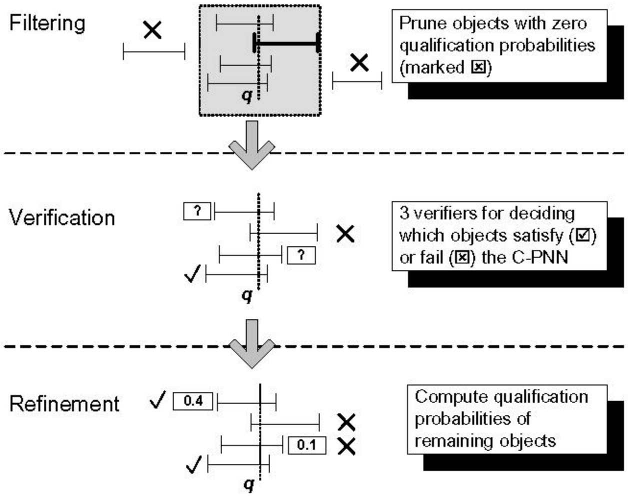

Filtering
q
Prune objects with zero qualification probabilities (marked ☒)
Verification
?
?
q
3 verifiers for deciding which objects satisfy (☑) or fail (☒) the C-PNN
Refinement
0.4
0.1
q
Compute qualification probabilities of remaining objects

Fig. 3. Solution Framework of C-PNN.

The C-PNN has another advantage: its answers can be more efficiently evaluated. In particular, we have developed *probabilistic verifiers* (or *verifiers* in short), which can make decisions on whether an object is included in the final answer, *without* computing the exact probability values. The verifiers derive a bound of qualification probabilities with algebraic operations, and test the bound with the threshold and tolerance constraints specified in the C-PNN. For example, a verifier may use the objects' uncertainty regions to deduce that the probability of object A in Figure 2 is less than 25%. If the threshold is 30% and the tolerance is 2%, we can immediately conclude that A must not be the answer, even though we may not know A's exact probability is 20%.

Figure 3 shows the role of verifiers in our solution, which consists of three phases. The first step is to prune or *filter* objects that must not be the nearest neighbor of q, using an R-tree based solution [8]. The objects with non-zero qualification probabilities (shaded) are then passed to the *verification* phase, where verifiers are used to decide if an object satisfies the C-PNN. In this figure, two objects have been determined in this stage. Objects that still cannot be determined are passed to the *refinement* phase, where the exact probability values are computed. We can see that the object with 0.4 probability is retained in the answer, while the other object (with 0.1 chance) is excluded.

In this paper, we focus on verification and refinement. We present three verifiers, which utilize an object's uncertainty information, as well as its relationship with other objects, in order to derive the lower and upper bounds of qualification probabilities. These verifiers can handle arbitrary pdfs. We also propose a paradigm to string these verifiers together in order to provide an efficient solution. Even if an object cannot be decided by the verifiers, the knowledge accumulated by the verifiers about the object can still be useful, and we show how this can facilitate the refinement process. As shown in the experiments, the price paid for using verifiers is justified by the lower cost of refinement. In some cases, the performance of the C-PNN has an order of magnitude of improvement in terms of execution time.

The rest of this paper is organized as follows. We discuss the related work in Section II. In Section III, we present the formal semantics of the C-PNN, and our solution framework. We discuss the details of verifiers in Section IV. Experimental results are described in Section V. We conclude the paper in Section VI. Appendix I describes the correctness proof for U-SR, one of the verifiers used in our solution.

II. Related Work

Recently, database systems for managing uncertainty have been proposed [10], [11], [12], [1]. Two major types of uncertainty are assumed in these works: tuple- and attribute-uncertainty. Tuple-uncertainty refers to the probability that a given tuple is part of a relation [10]. Attribute-uncertainty generally represents the inexactness in the attribute value as a range of possible values and a pdf bounded in the range [2], [3], [5], [13], [4]. Recently, a formal database model for attribute uncertainty has been proposed [14]. The imprecise data model studied here belongs to the attribute uncertainty.

An R-tree-based solution for PNN over attribute uncertainty has been presented in [8]. The main idea is to prune tuples with zero probabilities, using the fact that these tuples' uncertainty regions must not overlap with that of a tuple whose maximum distance from the query point is the minimum in the database. [5], [8] discuss the evaluation of qualification probabilities by transforming each uncertainty region into two functions: pdf and cdf of an object's distance from the query point. They show how this conversion can be done for 1D uncertainty (intervals) and 2D uncertainty (circle and line). The probabilities are then derived by evaluating an integral of an expression involving distance pdfs and cdfs from multiple objects. While our solution also uses distance pdfs and cdfs, its avoids a significant number of integration operations with the aid of verifiers.

Another method for evaluating a PNN is proposed in [9], where each object is represented as a set of points (sampled from the object's pdf). Compared with that work, our solution is tailored for a constrained version of PNN, where threshold and tolerance conditions are used to avoid computation of exact probabilities. Also, we do not need the additional work of sampling the pdf into points. Notice that this sampling process may introduce another source of error if there are not enough samples. In [15], a method for evaluating the probability that an object (represented as a histogram) is before another object in the time domain is presented. Their result could be adapted to answer a PNN that involves two objects, by viewing the time domain as the space domain. Our solution, on the other hand, addresses the PNN problem involving two or more objects.

Besides the PNN, the evaluation and indexing methods for other probabilistic queries have been studied. This includes range queries [16] and location-dependent queries [6]. The issues of uncertainty have also been considered in similarity matching in biometric databases [4].

III. An Evaluation Framework for C-PNN

Let us now present the semantics of the C-PNN (Section III-A). We then outline our solution in Section III-B.

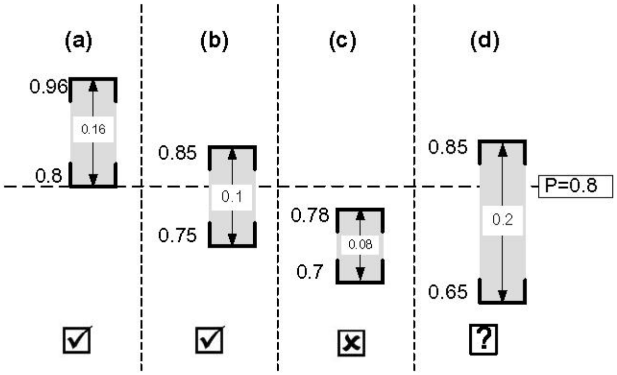


Fig. 4. C-PNN with $P = 0.8$ and $\Delta = 0.15$.

A. Definition of C-PNN

Let X be a set of uncertain objects in 1D space (i.e., an arbitrary pdf defined inside a closed interval), and X_i be the i^{th} object of X (where $i = 1, 2, \ldots, |X|$). We also suppose that $q \in \Re$ is the query point, and $p_i \in [0, 1]$ is the probability that X_i is the nearest neighbor of q (i.e., qualification probability). We call $p_i.l \in [0, 1]$ and $p_i.u \in [0, 1]$ the *lower* and *upper probability bound* of p_i respectively, such that $p_i.l \leq p_i.u$, and $p_i \in [p_i.l, p_i.u]$. In essence, $[p_i.l, p_i.u]$ is the range of possible values of p_i, and $p_i.u - p_i.l$ is the error in estimating the actual value of p_i. We denote the range $[p_i.l, p_i.u]$ as a *probability bound* of p_i.

Definition 1: A **Constrained Probabilistic Nearest Neighbor Query (C-PNN)** returns a set $\{X_i | i = 1, 2, \ldots, |X|\}$ such that p_i satisfies both of the following conditions:

- $p_i.u \geq P$
- $p_i.l \geq P$, or $p_i.u - p_i.l \leq \Delta$

where $P \in (0, 1]$ and $\Delta \in [0, 1]$.

Here P is called the *threshold* parameter. An object is allowed to be returned as an answer if its qualification probability is not less than P. Another parameter, called *tolerance*

(Δ), limits the amount of error allowed in the estimation of qualification probability p_i. Figure 4 illustrates the semantics of C-PNN. The probability bound $[p_j.l, p_j.u]$ of some object X_j (shaded) is shown in four scenarios. Let us assume that the C-PNN has a threshold $P = 0.8$ and tolerance $\Delta = 0.15$. Case (a) shows that the actual qualification probability p_j of some object X_j (i.e., p_j) is within a closed bound of $[p_j.l, p_j.u]$=$[0.8, 0.96]$. Since p_j must not be smaller than P, according to Definition 1, X_j is the answer to this C-PNN. In (b), X_j is also a valid answer since the upper bound of p_j (i.e., $p_j.u$) is equal to 0.85 and is larger than P. Moreover, the error of estimating p_j (i.e., 0.85-0.75), being 0.1, is less than $\Delta = 0.15$. Thus the two conditions of Definition 1 are satisfied. For case (c), X_j cannot be the answer, since the upper bound of p_j (i.e., 0.78) is less than P, and so the first condition of Definition 1 is violated. In (d), although object X_j satisfies the first requirement ($p_j.u = 0.85 \geq P$), the second condition is not met. According to Definition 1, it is not an answer to the C-PNN. However, if the probability bounds could later be "reduced" (e.g., by verifiers), then the conditions can be checked again. For instance, if $p_j.l$ is later updated to 0.81, then X_j will be the answer. Table I summarizes the symbols used in the definition of C-PNN.

Symbol	Meaning
X_i	Uncertain object i of a set X ($i = 1, 2, \ldots, \lvert X \rvert$)
q	Query point
p_i	Prob. that X_i is the NN of q (qualification prob.)
$[p_i.l, p_i.u]$	Lower & upper probability bounds
P	Threshold
Δ	Tolerance

TABLE I

SYMBOLS FOR C-PNN.

B. The Verification Framework

As shown in Figure 3, uncertain objects that cannot be filtered (shaded in Figure 3) require further processing. This set of unpruned objects, called the *candidate set* (or C in short), are passed to a *probabilistic verifier*, which reports a list of probability bounds of these objects. This list is sent to the *classifier*, which labels an object by checking its probability bounds against the definition of the C-PNN. In particular, an object is marked *satisfy* if it qualifies as an answer (e.g., Figures 4(a), (b)). It is labeled *fail* if it cannot satisfy the C-PNN (Figure 4(c)). Otherwise, the object is marked *unknown* (Figure 4(d)). This labeling process can be done easily by checking an object's probability bounds against the two conditions stated in Definition 1.

Figure 5 shows the three verifiers (namely RS, L-SR and U-SR, in shaded boxes), as well as the classifier. During initialization, all objects in the candidate set are labeled *unknown*, and their probability bounds are set to $[0, 1]$. Other information like the distance pdf and cdf is also precomputed for the candidate set objects. The candidate set is then passed to the first verifier (RS) for processing. The RS produces the newly computed probability bounds for the objects in the candidate sets, and sends this list to the classifier to label the objects. Any objects with the label *unknown* are transferred to the next verifier (L-SR) for another round of verification. The process goes on (with U-SR) until all the objects are either labeled *satisfy* or *fail*. When this happens, all the objects marked *satisfy* are returned to the user, and the query is finished. Thus, it is not always necessary for all verifiers to be executed.

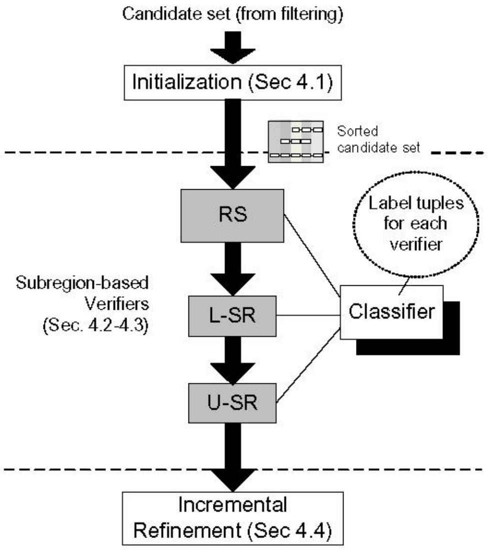
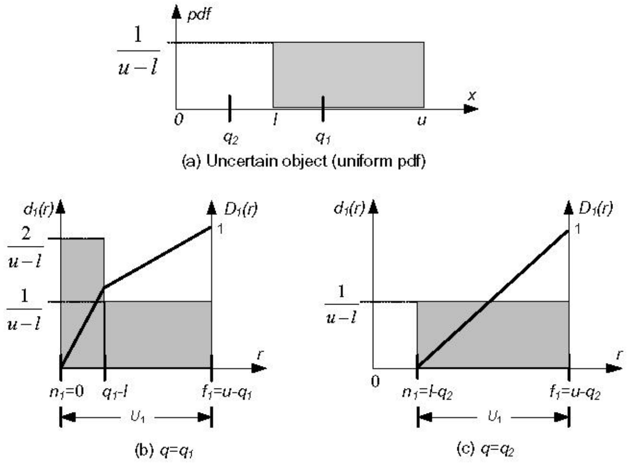

Candidate set (from filtering)
Initialization (Sec 4.1)
Sorted candidate set
RS
Label tuples for each verifier
Subregion-based Verifiers (Sec. 4.2-4.3)
L-SR
Classifier
U-SR
Incremental Refinement (Sec 4.4)

Fig. 5. The Verification Framework.

Notice that a verifier only adjusts the probability bound of an *unknown* object if this new bound is smaller than the one previously computed. Also, the verifiers are arranged in the order of their running times, so that if a low-cost verifier (e.g., the RS verifier) can successfully determine all the objects, there is no need to execute a more costly verifier (e.g., the L-SR verifier). In the end of verification, objects that are still labeled *unknown* are passed to the refinement stage for computing their exact probabilities. We discuss a faster technique based on the information obtained by the verifiers to improve this process in Section IV-D. Now let us examine the details of the verifiers.

IV. SUBREGION-BASED VERIFIERS

The verifiers presented here are collectively known as *subregion-based verifiers*, since the information of *subregions* is used to compute the probability bounds. A subregion is essentially a partition of the space derived from the uncertainty regions of the candidate set objects. Section IV-A discusses how subregions are produced. We then present the RS-verifier in Section IV-B, followed by the L-SR and U-SR verifiers in Section IV-C. In Section IV-D, we describe the "incremental refinement" method, which uses subregions to improve the refinement process.



(a) Uncertain object (uniform pdf)

(b) q=q_1

(c) q=q_2

Fig. 6. Distance pdf and cdf

A. Computing Subregion Probabilities

The initialization phase in Figure 5 performs two tasks: (1) computes the distance pdf and cdf for each object in the candidate set, and (2) computes *subregion probabilities*.

We start with the derivation of distance pdf and cdf. Let $R_i \in \Re$ be the absolute distance of an uncertain object X_i from q. That is, $R_i = |X_i - q|$. We assume that R_i takes on a value $r \in \Re$. Then, the distance pdf and cdf of X_i are defined as follows [5], [8]:

Definition 2: Given an uncertain object X_i, its **distance pdf**, denoted by $d_i(r)$, is a pdf of R_i; its **distance cdf**, denoted by $D_i(r)$, is a cdf of R_i.

Figure 6(a) illustrates an uncertain object X_1, which has a uniform pdf with a value of $\frac{1}{u-l}$ in an uncertainty region $[l, u]$. Two query points (q_1 and q_2) are also shown. Figure 6(b) shows the corresponding distance pdf (shaded) of $R_1 = |X_1 - q_1|$, with q_1 as the query point. Essentially, we derive the pdf of X_1's distance from q_1, which ranges from 0 to $u - q_1$. In $[0, q_1 - l]$, the distance pdf is obtained by summing up the pdf on both sides of q_1, which equals to $\frac{2}{u-l}$. The distance pdf in the range $[q_1 - l, u - q_1]$ is simply $\frac{1}{u-l}$. Figure 6(c) shows the distance pdf for query point q_2. For both queries, we draw the distance cdf in solid lines. Notice that the distance cdf can be found by integrating the corresponding distance pdf. From Figure 6, we observe that the distance pdf and cdf for the same uncertain object vary, and depend on the position of the query point. If the uncertainty pdf of X_i is in the form of a histogram (e.g., Figure 1(b)), its distance pdf/cdf can be found by first decomposing the histogram into a number of "histogram bars", where the pdf in the range of each bar is the same. We can then compute the distance pdf/cdf of each bar using the methods described in this paragraph, and combine the results to yield the distance pdf/cdf for the histogram.

We represent a distance pdf of each object as a histogram. The corresponding distance cdf is then a piecewise linear function. Note that although we focus on 1D uncertainty, our solution only needs distance pdfs and cdfs. Thus, our solution can be extended to 2D space, by computing the distance pdf and cdf from the 2D uncertainty regions, using the formulae discussed in [8].

Now, let us describe the definitions of near and far points of R_i, as defined in [5], [8]:

Definition 3: A **near point** of R_i, denoted by n_i, is the minimum value of R_i. A **far point** of R_i, denoted by f_i, is the maximum value of R_i.

We use U_i to denote the interval $[n_i, f_i]$. Figure 6(b) shows that when q_1 is the query point, $n_1 = 0$, $f_1 = u - q_1$, and $U_1 = [0, u - q_1]$. When q_2 is the query point, $U_1 = [l - q_2, u - q_2]$ (Figure 6(c)). We also let f_{min} and f_{max} be the minimum and maximum values of all the far points defined for the candidate set objects. We assume that the distance pdf of X_i has a non-zero value at any point in U_i.

Subregion Probabilities. Upon generating the distance pdfs and cdfs of the candidate set objects, the next step is to generate *subregions*. Let us first sort these objects in the ascending order of their near points. We also rename the objects as $X_1, X_2, \ldots, X_{|C|}$, where $n_i \leq n_j$ iff $i \leq j$. Figure 7(a) illustrates three distance pdfs with respect to a query point q, presented in the ascending order of their near points. The number above each range indicates the probability that an uncertain object has that range of distance from the query point.

In Figure 7(a), the circled values are called *end-points*. They include all the near points (e.g., e_1, e_2 and e_3), the minimum and maximum of far points (e.g., e_5 and e_6), and the point at which the distance pdf changes (e.g., e_4). No end points are defined between (e_1, e_2) and (e_5, e_6). We use e_j to denote the j-th end-point, where $j \geq 1$ and $e_j < e_{j+1}$. Moreover, $e_1 = n_1$.

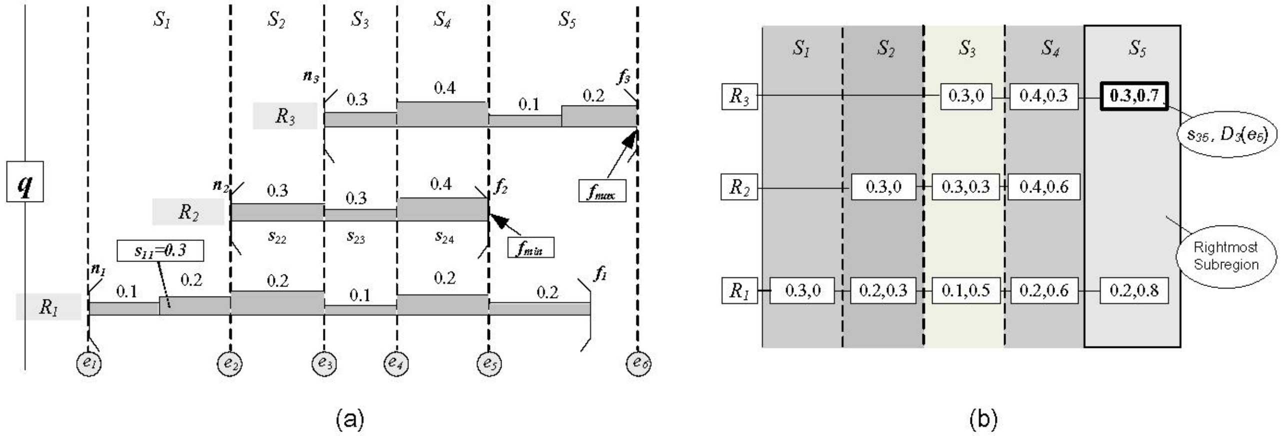
The adjacent pairs of end-points form the boundaries of a *subregion*. We label each subregion as S_j, where S_j is the interval $[e_j, e_{j+1}]$. Figure 7(a) shows five subregions, where $S_1 = [e_1, e_2]$, $S_2 = [e_2, e_3]$, and so on. The probability that R_i is located in S_j is called the *subregion probability*, denoted by s_{ij}. Figure 7(a) shows that $s_{22} = 0.3$, $s_{11} = 0.1 + 0.2 = 0.3$, and $s_{31} = 0$.

For each subregion S_j of an object X_i, we evaluate the subregion probability s_{ij}, as well as the distance cdf of S_j's lower end-point (i.e., $D_i(e_j)$). Figure 7(b) illustrates these pairs of values extracted from the example in (a). For example, for R_3 in S_5, the pairs $s_{35} = 0.3$ and $D_3(e_5) = 0.7$ are shown. These number pairs help the verifiers to develop the probability bounds. Table II presents the symbols used in our solution. Let us now examine how the verifiers work.

B. The Rightmost-Subregion Verifier

The **Rightmost-Subregion** (or RS) verifier uses the information in the "rightmost" subregion. In Figure 7(b), S_5 is the rightmost subregion. If we let $M \geq 2$ be the number of subregions for a given candidate set, then the following specifies an object's upper probability bound:

Lemma 1: The upper probability bound, $p_i.u$, is at most $1 - s_{iM}$, where s_{iM} is the probability that R_i is in S_M.



(a)

(b)

Fig. 7. Illustrating the distance pdfs and subregion probabilities.

Symbol	Meaning
C	$\{X_i \in X \| p_i > 0\}$ *(candidate set)*
R_i	$\|X_i - q\|$
$d_i(r)$	pdf of R_i *(distance pdf)*
$D_i(r)$	cdf of R_i *(distance cdf)*
n_i, f_i	Near and far points of distance pdf
U_i	The interval $[n_i, f_i]$
f_{min}, f_{max}	min. and max. of far points
e_k	The k-th end point
S_j	The j-th subregion, where $S_j = [e_j, e_{j+1}]$
M	Total no. of subregions
c_j	No. of objects with distance pdf in S_j
s_{ij}	$Pr(R_i \in S_j)$
q_{ij}	Qualification prob. of X_i, given $R_i \in S_j$
$[q_{ij}.l, q_{ij}.u]$	Lower & upper bounds of q_{ij}

TABLE II
SYMBOLS USED BY VERIFIERS.

The subregion S_M is the rightmost subregion. In Figure 7(b), $M = 5$. The upper bound of the qualification probability of object X_1, according to Lemma 1, is at most $1 - s_{15}$, or $1 - 0.2 = 0.8$.

To understand this lemma, notice that any object with distance larger than f_{min} *cannot* be the nearest neighbor of q. This is because f_{min} is the minimum of the far points of the candidate set objects. Thus, there exists an object X_k such that X_k's far point is equal to f_{min}, and that X_k is closer to q than any objects whose distances are larger than f_{min}. If we also know the probability of an object located beyond a distance of f_{min} from q, then its upper probability bound can be deduced. For example, Figure 7(a) shows that the distance of X_1 from q (i.e., R_1) has a 0.2 chance of being more than f_{min}. Thus, X_1 is not the nearest neighbor of q with a probability of at least 0.2. Equivalently, the upper probability bound of X_1, i.e., $p_1.u$, is $1 - 0.2 = 0.8$. Note that 0.2 is exactly the probability that R_1 lies in the rightmost subregion S_5, i.e., s_{15}, and thus $p_1.u$ is equal to $1 - s_{15}$. This result can be generalized for any object in the candidate set, as shown in Lemma 1.

Notice that the RS verifier only handles the objects' upper probability bounds. To improve the lower probability bound, we need the L-SR verifier, as described next.

C. The Lower-Subregion and Upper-Subregion Verifiers

The second type of verifiers, namely the Lower-Subregion (**L-SR**) and Upper-Subregion (**U-SR**) Verifiers, uses subregion probabilities to derive the objects' probability bounds. For each subregion the L-SR (U-SR) verifier computes the lower (upper) probability bound of each object.

We define the term *subregion qualification probability* (q_{ij} in short), which is the chance that X_i is the nearest neighbor of q, given that its distance from q, i.e., R_i, is inside subregion S_j. We also denote the lower bound of the subregion qualification probability as $q_{ij}.l$. Our goal is to derive $q_{ij}.l$ for object X_i in subregion S_j. Then, the lower probability bound of X_i, i.e., $p_i.l$, is evaluated. Suppose there are $c_j (c_j \geq 1)$ objects with non-zero subregion probabilities in S_j. For example, $c_3 = 3$ in Figure 7(a), where all three objects have non-zero subregion probabilities S_3. The following lemma is used by the L-SR verifier to compute $q_{ij}.l$.

Lemma 2: Given an object $X_i \in C$, if $e_j \leq R_i \leq e_{j+1}$ $(j = 1, 2, \ldots, M-1)$, then

$$q_{ij}.l = \begin{cases} \frac{1}{c_j} \prod_{U_k \cap S_j \neq \emptyset \wedge k \neq i} (1 - D_k(e_j)) & \text{if } c_j > 1 \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

In words, Lemma 2 calculates $q_{ij}.l$ for object X_i by multiplying the expressions of distance cdfs for all objects with non-zero subregion probabilities in S_j. We will prove this lemma in Section IV-C.1. To illustrate the lemma, Figure 7(a) shows that $q_{11}.l$ (for X_1 in subregion S_1) is equal to 1, since $c_1 = 1$. On the other hand, $q_{23}.l$ (for X_2 in S_3) is $\frac{(1-0.5)(1-0)}{3}$, or 0.167.

Next, we define a real constant Y_j, where

$$Y_j = \prod_{U_k \cap S_j \neq \emptyset} (1 - D_k(e_j)) \quad (2)$$

Then, Equation 1 can be rewritten as:

$$q_{ij}.l = \frac{Y_j}{c_j \cdot (1 - D_i(e_j))} \quad (3)$$

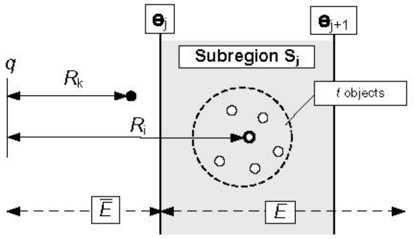


Fig. 8. Correctness proof of L-SR.

By computing Y_j first, the L-SR can use Equation 3 to compute $q_{ij}.l$ easily for each object in the same subregion S_j.

After the values of $q_{ij}.l$ have been obtained, the lower probability bound $(p_i.l)$ of object X_i can be evaluated by:

$$p_i.l = \sum_{j=1}^{M-1} s_{ij} \cdot q_{ij}.l \tag{4}$$

The product $s_{ij} \cdot q_{ij}.l$ is the minimum qualification probability of X_i in subregion s_{ij}, and Equation 4 is the sum of this product over the subregions. Note that the rightmost subregion (S_M) is not included, since the probability of any object S_M must be zero.

We also state the main result about U-SR, which evaluates the upper subregion probability bound:

$$q_{ij}.u = \frac{1}{2} \cdot \Big(\prod_{U_k \cap S_{j+1} \neq \emptyset \wedge k \neq i} (1 - D_k(e_{j+1})) + \prod_{U_k \cap S_j \neq \emptyset \wedge k \neq i} (1 - D_k(e_j)) \Big) \tag{5}$$

The upper probability bound $(p_i.u)$ can be computed by replacing $q_{ij}.l$ with $q_{ij}.u$ in Equation 4. The proof of Equation 5 can be found in Appendix I. Next, we present the proof of Lemma 2 for L-SR.

1) Correctness Proof of the L-SR Verifier: We first state a claim about subregions: if there exists a set K of objects whose distances from q (i.e., R_i) are certainly inside a subregion S_j, and all other objects $(C-K)$'s distances are in subregions $j+1$ or above, then the qualification probability of each objects in K is equal to $\frac{1}{|K|}$. This is because all objects in $C-K$ cannot be the nearest neighbor of q, and all objects in K must have the same qualification probability. In Figure 7(a), for example, if the distances R_1 and R_2 are inside $S_2 = [e_2, e_3]$, then $p_1 = p_2 = \frac{1}{2}$, and $p_3 = 0$. The following lemma states this formally.

Lemma 3: Suppose there exists a nonempty set $K (K \subseteq C)$ of objects such that $\forall X_i \in K, e_j \leq R_i \leq e_{j+1}$. If $\forall X_m \in C - K, R_m > e_{j+1}$, then for any $X_i \in K$, $p_i = \frac{1}{|K|}$, where $|K|$ is the number of objects in K.

The formal proof of Lemma 3 can be found in the technical report [17]. This lemma by itself can be used to obtain the qualification probabilities for the scenario when there is only one subregion (i.e., $M = 1$). Here, the distances of all the objects in the candidate set C from q are located in one subregion, S_1. Using Lemma 3, we obtain $p_i = \frac{1}{|C|}, \forall X_i \in C$.

We can now prove Lemma 2. Let us examine when c_j, the number of objects with non-zero subregion probabilities in subregion S_j, is equal to 1. In fact, this scenario happens to subregion S_1, i.e., $j = 1$, since only this region can accommodate a single distance pdf (e.g., $d_1(r)$ in Figure 7). If we also know that distance R_i is in subregion S_j, then X_i must be the nearest neighbor. Thus, the lower subregion qualification probability bound $q_{ij}.l$ is equal to 1, as stated in the lemma.

For the case $c_j > 1$, we derive the subregion qualification probability, q_{ij}. Let E denote the event that "all objects in the candidate set have their actual distances from q not smaller than e_j". Also, let $\bar{E}$ be the complement of event E, i.e., "there exists an object whose distance from q is less than e_j". Figure 8 illustrates these two events. If $Pr(E)$ denotes the probability that event E is true, then $Pr(E) = 1 - Pr(\bar{E})$. Let N be the event "Object X_i is the nearest neighbor of q". Then, using the law of total probability, we have:

$$q_{ij} = Pr(N|E) \cdot Pr(E) + Pr(N|\bar{E}) \cdot Pr(\bar{E}) \tag{6}$$

If $\bar{E}$ is true, there is at least one object X_k whose distance R_k is not larger than e_j (Figure 8). Since $R_k < R_i$, object X_k must be closer to q than object X_i. Consequently, $Pr(N|\bar{E}) = 0$, and Equation 6 becomes:

$$q_{ij} = Pr(N|E) \cdot Pr(E) \tag{7}$$

If E is true, the distances of all objects from q are beyond e_j. Suppose there are t objects (including X_i) such that their distances are in S_j (Figure 8). Using Lemma 3, we obtain $Pr(N|E) = \frac{1}{t}$. Since $t \leq c_j$ (where c_j is the number of objects with non-zero subregion probabilities in S_j), we have

$$Pr(N|E) \geq \frac{1}{c_j} \tag{8}$$

To obtain $Pr(E)$, note that the probability that an object X_k's distance is e_j or more is simply $1 - D_k(e_j)$. We then multiply all these probabilities, as $\prod_{R_k \geq e_j \wedge k \neq i} (1 - D_k(e_j))$. This can be simplified to:

$$Pr(E) = \prod_{U_k \cap S_j \neq \emptyset \wedge k \neq i} (1 - D_k(e_j)) \tag{9}$$

since any object whose subregion probability is zero in S_j must have the distance cdf at e_j, i.e.,$D_k(e_j)$ equal to zero. Combining Equations 7, 8 and 9, we can obtain the lower bound of q_{ij}, i.e., $q_{ij}.l$, as stated in Lemma 2.

D. Incremental Refinement

As discussed in Section III-B, some objects may still be unclassified after all verifiers have been applied. The exact probabilities of these objects must then be computed or "refined'. This can be expensive, since numerical integration has to be performed over the object's distance pdf [5]. This process can be performed faster by using the information provided by the verifiers. Particularly, the probability bounds of each object in each subregion (i.e., $[q_{ij}.l, q_{ij}.u]$) have already been computed by the verifiers. Therefore, we can decompose the refinement of qualification probability into a series of probability calculations inside subregions. Once we have computed the probability q_{ij} for subregion S_j, we collapse $[q_{ij}.l, q_{ij}.u]$ into a single value q_{ij}, update the probability bound $[p_i.l, p_i.u]$, and test this new bound with classifier. We repeat this process with another subregion until we can classify the object. This "incremental refinement" scheme is usually faster than directly

Algorithm	Qualification Prob. Bound	Cost
RS	Upper	$O(\|C\|)$
L-SR	Lower	$O(\|C\|M)$
U-SR	Upper	$O(\|C\|M)$

TABLE III
COMPLEXITY OF VERIFIERS.

computing qualification probabilities, since checking with a classifier is cheap, and performing numerical integration on a subregion is faster than on the whole uncertainty region, which has a larger integration area than a subregion. The formulae of this method can be found in [17].

We complete this section with a discussion on the implementation issues. We store the subregion probabilities (s_{ij}) and the distance cdf values $(D_i(e_j))$ for all objects in the same subregion as a list. These lists are indexed by a hash table, so that the information of each subregion can be accessed easily. The space complexity of this structure is $O(|C|M)$. It can be extended to a disk-based structure by partitioning the lists into disk pages. The complexities of the verifiers are shown in Table III. The three verifiers, as shown in Figure 5, are arranged in the ascending order of these running costs. The complexity of verification (including initialization and sorting of candidate set objects) is $O(|C|(\log |C| + M))$, and is lower than the evaluation of exact probabilities $(O(|C|^2 M))$. The derivation of these costs can be found in [17].

V. EXPERIMENTAL RESULTS

We have performed experiments to examine our solution. We present the simulation setup in Section V-A, followed by the results in Section V-B.

A. Experimental Setup

We use the *Long Beach* dataset[1], where the 53,144 intervals, distributed in the x-dimension of 10K units, are treated as uncertainty regions with uniform pdfs. For each C-PNN, the default values of threshold (P) and tolerance (Δ) are 0.3 and 0.01 respectively. We suppose a user of the C-PNN is not interested in small probabilities, by assuming $P > 0.1$. The query points are randomly generated. Each point in the graph is an average of the results for 100 queries.

We compare three strategies of evaluating a C-PNN. The first method, called *Basic*, evaluates the exact qualification probabilities using the formula in [5]. The second one, termed *VR*, uses probabilistic verifiers and incremental refinement. The last method (*Refine*) skips verification and performs incremental refinement directly. All these strategies assume the candidate set is ready i.e., filtering has already been applied to the original dataset. On average, the candidate set has 96 objects.

The experiments, written in Java, are executed on a PC with an Intel T2400 1.83GHz CPU and 1024MB of main memory. We have also implemented the filtering phase by using the R-tree library in [18].

B. Results

1. Cost of the Basic Method. We first compare the time spent on the *Basic* with filtering. Figure 9 shows that the fraction of total time spent in these two operations on synthetic data sets with different data set sizes. As the total table size $|T|$ increases, the time spent on the *Basic* solution increases more than filtering, and so its running time starts to dominate the filtering time when the data set size is larger than 5000. As we will show next, other methods can alleviate this problem.

2. Effectiveness of Verification. In Figure 10, we compare the time required by the three evaluation strategies under a wide range of values of P. Both *Refine* and *VR* perform better than *Basic*. At $P = 0.3$, for instance, the costs for *Refine* and *VR* are 80% and 16% of *Basic* respectively. The reason is that both techniques allow query processing to be finished once all objects have been determined, without waiting for the exact qualification probabilities to be computed. For large values of P, most objects can be classified as *fail* quickly when their upper probability bounds are detected to be lower than P. Moreover, *VR* is consistently better than *Refine*; it is five times faster than *Refine* at $P = 0.3$, and 40 times faster at $P = 0.7$. This can be explained by Figure 11, which shows the execution time of filtering, verification and refinement for *VR*. While the filtering time is fixed, the refinement time decreases with P. The verification takes only 1ms on average, and it significantly reduces the number of objects to be refined. In fact, when $P > 0.3$, no more qualification probabilities need to be computed. Thus, *VR* produces a better performance than *Refine*.

3. Comparison of Verifiers. Figure 12 shows the fraction of objects labeled *unknown* after the execution of verifiers in the order: {RS, L-SR, U-SR}. This fraction reflects the amount of work needed to finish the query. At $P = 0.1$, about 75% of *unknown* objects remain after the RS is finished; 7% more objects are removed by L-SR; 15% *unknown* objects are left after the U-SR is executed. When P is large, RS and U-SR perform better, since they reduce upper probability bounds, so that the objects have a higher chance of being labeled as *fail*. L-SR works better for small P (as seen from the gap between the RS and L-SR curves). L-SR increases the lower probability bound, so that an object is easier to be classified as *satisfy* at small P. In this experiment, U-SR performs better than L-SR. This is because the candidate set size is large (about 96 objects), so that the probabilities of the objects are generally quite small. Since U-SR reduces their upper probability bounds, they are relatively easy to be verified as *fail*, compared with L-SR, which attempts to raise their lower probability bounds.

4. Effect of Tolerance. Next, we measure the fraction of queries finished after verification under different tolerance. Figure 13 shows that as Δ increases from 0 to 0.2, more queries are completed. When $\Delta = 0.16$, about 10% more queries will be completed than when $\Delta = 0$. Thus, the use of tolerance can improve query performance.

5. Gaussian pdf. Finally, we examine the effect of using

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

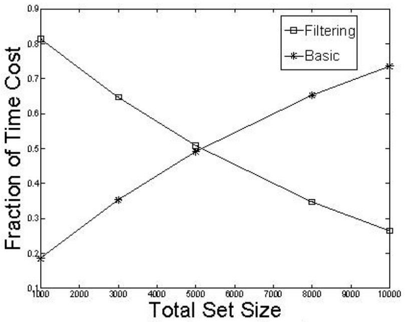


Fig. 9. Basic vs. Filtering.

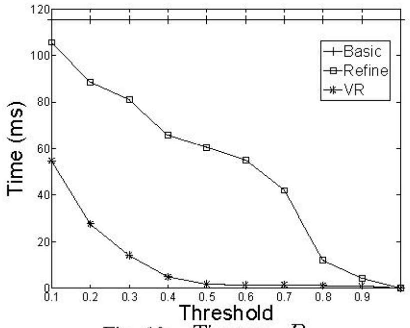


Fig. 10. Time vs. P.

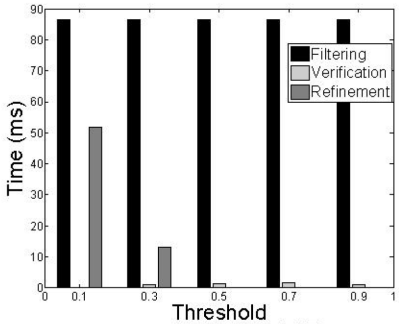


Fig. 11. Analysis of *VR*.

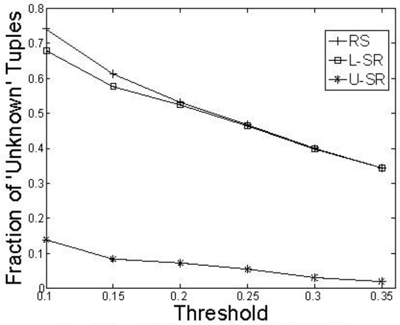


Fig. 12. RS, L-SR,and U-SR.

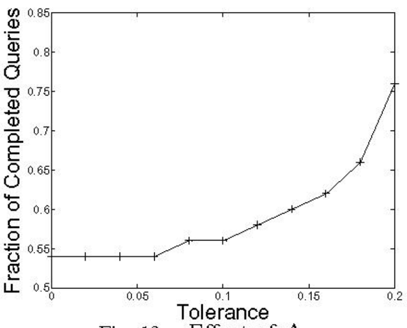


Fig. 13. Effect of Δ.

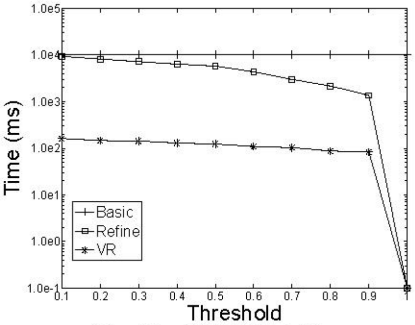


Fig. 14. Gaussian pdf.

a Gaussian distribution as the uncertainty pdf for each object. Each Gaussian pdf, approximated by a 300-bar histogram, has a mean at the center of its range, and a standard deviation of 1/6 of the width of the uncertainty region. Figure 14 shows the time drawn in log scale. *VR* again outperforms the other two methods. The saving is more significant than when uniform pdf is used. This is because the probability evaluation of Gaussian pdf is expensive, but this operation can be effectively avoided by the verifiers. This experiment shows that our method also works well with Gaussian pdf. The little time cost for both Refine and VR at threshold $P = 1$ is due to the fact that only one candidate, if any, can satisfy the query at $P = 1$. By checking against these conditions, both methods can accept or reject candidate objects with ease.

VI. Conclusions

Uncertainty management has recently attracted a lot of research interest. In this paper, we studied the evaluation of a C-PNN query on uncertain data. By using threshold and tolerance constraints, a C-PNN provides users with more flexibility in controlling the confidence and quality of their answers. Moreover, by evaluating C-PNN with the help of probabilistic verifiers, the problem of high costs for computing exact probabilities can be alleviated. These verifiers allow answers to be quickly determined, by using the different properties of subregions to compute the probability bounds. For future work, we will investigate other kinds of verifiers, and study the evaluation of k-NN queries.

Acknowledgements

The work described in this paper was supported by the Research Grants Council of the Hong Kong SAR, China (Project No. PolyU 5138/06E). We would like to thank the anonymous reviewers for their insightful comments and suggestions.

References

[1] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, “Model-driven data acquisition in sensor networks,” in *Proc. VLDB*, 2004.

[2] P. A. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, “Querying the uncertain position of moving objects,” in *Temporal Databases: Research and Practice*, 1998.

[3] D.Pfoser and C. Jensen, “Capturing the uncertainty of moving-objects representations,” in *Proc. SSDBM*, 1999.

[4] C. Böhm, A. Pryakhin, and M. Schubert, “The gauss-tree: Efficient object identification in databases of probabilistic feature vectors,” in *Proc. ICDE*, 2006.

[5] R. Cheng, D. Kalashnikov, and S. Prabhakar, “Evaluating probabilistic queries over imprecise data,” in *Proc. ACM SIGMOD*, 2003.

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

[7] M. Mokbel, C. Chow, and W. G. Aref, “The new casper: Query processing for location services without compromising privacy,” in *VLDB*, 2006.

[8] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, “Querying imprecise data in moving object environments,” *IEEE TKDE*, vol. 16, no. 9, Sept. 2004.

[9] H. Kriegel, P. Kunath, and M. Renz, “Probabilistic nearest-neighbor query on uncertain objects,” in *DASFAA*, 2007.

[10] N. Dalvi and D. Suciu, “Efficient query evaluation on probabilistic databases,” in *Proc. VLDB*, 2004.

[11] O. Mar, A. Sarma, A. Halevy, and J. Widom, “ULDBs: databases with uncertainty and lineage,” in *VLDB*, 2006.

[12] C. Mayfield, S. Singh, R. Cheng, and S. Prabhakar, “Orion: A database system for managing uncertain data, ver. 0.1 (http://orion.cs.purdue.edu),” 2006.

[13] V. Ljosa and A. K. Singh, “APLA: Indexing arbitrary probability distributions,” in *Proc. ICDE*, 2007.

[14] Singh et al, “Database support for pdf attributes,” in *Proc. ICDE*, 2008.

[15] C. Dyreson and R. Snodgrass, “Supporting valid-time indeterminacy,” *ACM Trans. Database Syst.*, vol. 23, no. 1, 1998.

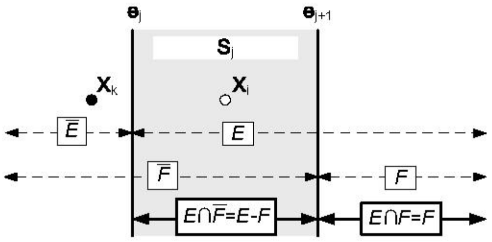


Fig. 15. Correctness proof of U-SR.

[16] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar, "Indexing multi-dimensional uncertain data with arbitrary probability density functions," in *Proc. VLDB*, 2005.

[17] R. Cheng, J. Chen, M. Mokbel, and C. Chow, "Efficient processing of probabilistic nearest-neighbor queries over uncertain data (technical report)," 2007. [Online]. Available: http://www.comp.polyu.edu.hk/˜csckcheng/tech/pnnq.pdf

[18] M.Hadjieleftheriou, "Spatial index library version 0.44.2b." [Online]. Available: http://u-foria.org/marioh/spatialindex/index.html

Appendix I
Correctness of the U-SR Verifier

We now prove that Equation 5 is correct.

Proof: First, let F be the event "$\forall T_k \in C$, where $k \neq i$, $X_k \geq e_{j+1}$", and $\bar{F}$ be the event "$\exists T_k \in C$ s.t. $k \neq i \wedge X_k < e_{j+1}$". Again, we let N be the event "T_i is NN of q". Since F and $\bar{F}$ are mutually exclusive, $Pr(F) = 1 - Pr(\bar{F})$. Using the definition of E defined in Section IV-C, we can rewrite Equation 7 as:

$$q_{ij} = Pr(N|E\cap F)\cdot Pr(E\cap F) + Pr(N|E\cap\bar{F})\cdot Pr(E\cap\bar{F}) \quad (10)$$

Figure 15 illustrates the relationship between the events E, $\bar{E}$, F and $\bar{F}$. Each dotted line associated with each event represents the possible values of X_k's if that event happens.

If $E \cap F$ is true, then all tuples except T_i have their X_i's values not smaller than e_{j+1}. Since $X_i \leq e_{j+1}$, it must be the nearest neighbor. Thus, $Pr(N|E \cap F) = 1$.

Next, suppose $E\cap\bar{F}$ is true. Then, in addition to T_i, $m \geq 1$ other tuple(s) is (are) on the left of e_{j+1}. Since E is also true, the values of X_k for all these m tuples must also be in in S_j. Using Lemma 3, we can then deduce that $Pr(N|E \cap \bar{F}) = \frac{1}{m+1}$. The maximum value of $Pr(N|E\cap\bar{F})$ is $\frac{1}{2}$, which happens when $m = 1$.

Now, $Pr(E \cap F) = Pr(F)$ since $F \subset E$ (Figure 15). Also, $Pr(F) = \prod_{X_k \geq e_{j+1} \wedge k \neq i}(1 - D_k(e_{j+1}))$, which can be simplified as $\prod_{U_k \cap S_{j+1} \neq \emptyset \wedge k \neq i}(1 - D_k(e_{j+1}))$, since any tuple T_k whose uncertainty region does not fall into S_{j+1} must have $D_k(e_{j+1})$ equal to zero. Moreover, $Pr(E \cap \bar{F}) = Pr(E) - Pr(F)$ (Figure 8), with $Pr(E)$ given by Equation 9. By substituting these expressions into Equation 10, we can obtain the expression of $q_{ij}.u$ as stated in Equation 5. ■

Notice that Equation 5 can be written as:

$$q_{ij}.u = \frac{1}{2}(\frac{Y_j}{1 - D_i(e_j)} + \frac{Y_{j+1}}{1 - D_i(e_{j+1})}) \quad (11)$$

where Y_j and Y_{j+1} are given by Equation 2. Hence, similar to L-SR, $q_{ij}.u$ can be obtained easily. If both Y_j and Y_{j+1} have been stored (e.g., in a 1D array) after the evaluation of the L-SR, their values can then be reused by U-SR. We thus obtain $p_i.u$ in $O(|C|M)$ times. The detailed derivation of this cost can be found in [17].