Probabilistic Group Nearest Neighbor Queries in Uncertain Databases

Xiang Lian, *Student Member*, *IEEE*, and Lei Chen, *Member*, *IEEE*

Abstract—The importance of query processing over uncertain data has recently arisen due to its wide usage in many real-world applications. In the context of uncertain databases, previous works have studied many query types such as *nearest neighbor query*, *range query*, *top-k query*, *skyline query*, and *similarity join*. In this paper, we focus on another important query, namely, *probabilistic group nearest neighbor* (PGNN) *query*, in the uncertain database, which also has many applications. Specifically, given a set, Q, of query points, a PGNN query retrieves data objects that minimize the aggregate distance (e.g., sum, min, and max) to query set Q. Due to the inherent uncertainty of data objects, previous techniques to answer *group nearest neighbor* (GNN) *query* cannot be directly applied to our PGNN problem. Motivated by this, we propose effective pruning methods, namely, *spatial pruning* and *probabilistic pruning*, to reduce the PGNN search space, which can be seamlessly integrated into our PGNN query procedure. Extensive experiments have demonstrated the efficiency and effectiveness of our proposed approach, in terms of the *wall clock time* and the *speed-up ratio* against *linear scan*.

Index Terms—Probabilistic group nearest neighbor queries, uncertain database.

1 Introduction

Recently, query processing over uncertain data has drawn much attention from the database community, due to its wide usage in many applications such as sensor network monitoring [8], object identification [1], and moving object search [3], [2], [15]. Many real-world application data inherently contain uncertainty. For example, sensor data collected from different sites may be distorted for various reasons like the environmental factors, device failures, or battery power. Moreover, in the mobile environment, the positions of mobile users stored in the database may deviate from their actual values at the query time due to the precision of positioning devices, transmission delay, and so on. Therefore, in these situations, each object can be modeled as a so-called *uncertainty region* [3], [24], instead of a precise point. For simplicity, in this paper, we assume that the uncertainty region is of hypersphere shape [3], [24]. Fig. 1 illustrates a 2D example of uncertain database, where each uncertain object can locate within a circle with arbitrary distribution.

While many proposed techniques for answering queries (e.g., *nearest neighbor (NN) query* and *range query*) assume that data objects are precise, they cannot be directly applied to handle uncertain data (otherwise, inaccuracy or even errors may be introduced). Thus, it is crucial to design novel approaches to efficiently and accurately answer queries over uncertain objects. In the context of uncertain databases, previous works have studied query types such as *range query* [4], [5], [24], *NN query* [3], [4], [14], *top-k query* [20], [23], *skyline query* [19], and *similarity join* [13].

In this paper, we focus on another important type of query, namely, *GNN query* [17], [18], in uncertain databases, which, to the best of our knowledge, no other work has studied before. Specifically, in a "certain" database $\mathcal{D}$ (containing precise data objects), given a set of query points, $Q = \{q_1, q_2, \ldots, q_n\}$, a GNN query [17], [18] retrieves one data object $o \in \mathcal{D}$ that minimizes an aggregate distance function $adist_f(o, Q)$, where $adist_f(o, Q)$ is defined as a monotonically increasing function $f(..)$[1] with respect to object o and query set Q. That is,

$$adist_f(o, Q) = f(dist(o, q_1), dist(o, q_2), \ldots, dist(o, q_n)),$$

where $dist(x, y)$ is the *euclidean distance* between two data objects x and y. As an example, for sum aggregate distance function, we have $adist_{sum}(o, Q) = \sum_{i=1}^{n} dist(o, q_i)$. In the sequel, we refer to $adist_f(o, Q)$ as $adist(o, Q)$ for brevity unless specific aggregate function f is used.

In an uncertain database, however, each data object has "dynamic" attributes, which means that the value of an attribute locates within a range with some probability. Therefore, the pairwise distance between any two objects is no longer a constant; instead, it is a variable. Correspondingly, we have to redefine the GNN query over uncertain objects. In particular, in an uncertain database, a *probabilistic group nearest neighbor* (PGNN) query retrieves a set of uncertain objects such that their probability of being GNN is greater than a user-specified probability threshold α, where $\alpha \in (0, 1]$.

The PGNN query is important in many applications. For instance, in a forest where many sites are on fire, the exact

• *The authors are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China.*
E-mail: {xlian, leichen}@cse.ust.hk.

Manuscript received 1 Aug. 2007; revised 27 Nov. 2007; accepted 14 Jan. 2008; published online 13 Feb. 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-08-0394.
Digital Object Identifier no. 10.1109/TKDE.2008.41.

1. Function f is monotonically increasing iff: $\forall i$, $x_i \geq x_i' \mapsto f(x_1, \ldots, x_n) \geq f(x_1', \ldots, x_n')$.

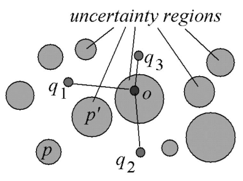


Fig. 1. A group nearest neighbor (GNN) query in the uncertain database.

location of each site on fire is not precise (e.g., due to wind or moving speed of the fire), which can thus be modeled as an uncertain object. Several firefighters located at different places want to get together so as to put out fires (assuming a single firefighter cannot put out fire alone). In this case, they can issue a PGNN query (with locations of firefighters as query points) to find a site that minimizes either their total traveling distance (sum aggregate distance) or minimum (maximum) time (traveling distance) for them to reach the site (i.e., min (max) aggregate distance). In an image database, the stored images may contain noises due to the inaccuracy of capturing devices or some environmental factors. Thus, the feature vectors extracted from these images can be considered as uncertain data, which means that each real feature value locates within a range. Furthermore, due to the gap between semantic concepts and low-level image features [22], a single image is often not enough to express a semantic concept that users want to query. In this case, we can select a group of example query images and issue a PGNN query to find images that have the minimum sum aggregate distance to the query set. For instance, if users want to search all the images about "sunset," they can select a set of query images with different sunset scenes, such as sunset in mountains, sea, and grassland, and conduct a PGNN query over uncertain image feature vectors in the database.

The PGNN query can be also used when a military unit wants to find moving enemy that minimizes the maximum distances to troops at different locations. As a meteorology example, consider a system that monitors severe weather phenomena (e.g., typhoons). We can issue a PGNN query to identify ships or moving vehicles that are under the highest potential danger based on their distances to any phenomenon and warn them as early as possible. In addition to stand-alone methods, PGNN can be integrated into other related problems such as k-medoids or outlier detection [17], where the underlying data are imprecise.

Motivated by the fact that previous methods can only handle the GNN query over precise data [17], in this paper, we propose efficient and effective approaches to address PGNN queries over uncertain databases.

Specifically, we make the following contributions:

1. We formalize, in Section 3, the problem of PGNN *query* over the uncertain database.
2. We propose, in Section 4, the effective pruning methods, namely *spatial pruning* and *probabilistic pruning*, to reduce the PGNN search space which can be seamlessly integrated into a general framework for answering PGNN queries discussed in Section 5.
3. We generalize, in Section 6, our proposed solutions to answer PGNN queries not only with precise query points but also uncertain query objects.
4. We demonstrate through extensive experiments, in Section 8, the effectiveness of the pruning methods as well as the efficiency of our proposed PGNN query procedure.

In addition, Section 2 briefly overviews previous methods to retrieve GNN over precise data objects and related work on query processing in uncertain databases. Section 7 discusses variants of the PGNN query. Finally, Section 9 concludes this paper.

2 Related Work

Section 2.1 reviews previous methods to answer GNN queries in the "certain" database. Section 2.2 illustrates query processing over uncertain databases.

2.1 Group Nearest Neighbor (GNN) Queries

GNN was first proposed by Papadias et al. [17], which retrieves data objects in the database that minimize their sum aggregate distances to a user-specified set of query points. Specifically, given a database $\mathcal{D}$ and a set, Q, of n query points, $q_1, q_2, \ldots$ and q_n, a GNN query obtains a data object o that minimizes the sum distance from o to the query set Q, that is, $\sum_{i=1}^{n} dist(o, q_i)$, where $dist(x, y)$ is the *euclidean distance* between two data objects x and y. Later, the authors [18] generalized sum to any monotonically increasing aggregate distance function f, in which they also demonstrated how min and max aggregate distance functions can be solved in the same framework for GNN. Note that, GNN query with aggregate distance function is also called *aggregate nearest neighbor* (ANN) query in [18]. Throughout this paper, however, we always name this kind of query (with aggregates) as GNN, unless otherwise specified.

Papadias et al. [17] proposed three methods to answer GNN queries with the assumption that all the data objects in the database are indexed by a multidimensional structure, R-tree [9]. The first proposed approach, namely, *multiple query method* (MQM), applies the *threshold algorithm* [7] to retrieve the top-1 data point that minimizes the score (i.e., the sum distance to the query set Q). Specifically, MQM traverses the R-tree index in either depth-first or best-first manner to incrementally obtain NNs of each query point. The algorithm also maintains a global threshold, $best\_dist$, to record the summation of distance $dist(q_i, cur\_NN(q_i))$ (for $1 \le i \le n$) from each query point q_i to its current NN $cur\_NN(q_i)$ that is visited. MQM accesses NNs of query points in a round-robin fashion. Each time when a data point o is retrieved, the sum aggregate distance $dist(o, Q)$ from o to query set Q is computed. In case the minimum sum distance among all the retrieved data points so far is below the threshold $best\_dist$, the procedure MQM can terminate, since those data objects that have not been accessed would not have the sum distance smaller than $best\_dist$.

The second approach, namely, the *single point method* (SPM), calculates the centroid q of query set Q and uses q to prune data objects/nodes when traversing the R-tree. In particular, it computes the lower bound of the sum aggregate distance from object/node to query set via q using the *triangle inequality*. If the lower bound is larger than or equal to the smallest sum distance $best\_dist$ among objects we have visited so far, then we can safely prune this object/node.

The third approach, namely, *minimum bounding method* (MBM) uses a *minimum bounding rectangle* (MBR, denoted as $\mathrm{MBR}(Q)$) to bound all the query points in Q. Then, $\mathrm{MBR}(Q)$ is considered as the representative of query points and can be applied to the query procedure to facilitate pruning data points/node MBRs. Similar to SPM, the lower bound distance between $\mathrm{MBR}(Q)$ and any data point/node MBR is calculated. If this lower bound is larger than or equal to the minimum distance $best\_dist$ among data points that have been found so far, then the data point/node MBR can be safely pruned.

The experimental results of GNN queries reported in [18] indicate that SPM and MBM outperform MQM. This is reasonable, since SPM and MBM traverse R-tree for only one pass (i.e., each node is accessed, at most, once), whereas MQM has to find NNs for query points by traversing the R-tree for multiple times.

2.2 Query Processing in Uncertain Databases

Query processing over uncertain data has attracted much attention from the database community due to its importance in many applications [8], [1], [3], [2], [15], [21], [23], where real-world data inherently contain uncertainty themselves. Since traditional approaches for answering queries always assume that data are precise, they are not directly applicable to handle uncertainty. Therefore, many techniques have been designed specifically for uncertain databases, with queries such as *range query* [4], [5], [24], NN *query* [3], [4], [14], *top-k query* [20], [23], *skyline query* [19], and *similarity join* [13]. To the best of our knowledge, so far, no existing work has studied the GNN query in the context of the uncertain database, which assumes that data objects can have "dynamic" attributes in the data space (i.e., locating anywhere within the uncertainty regions). Due to the inherent uncertainty in many real-world data from various applications, we need to find a solution that can efficiently answer the GNN query in uncertain databases.

The most related work to our GNN problem in the uncertain database is the *probabilistic nearest neighbor query* (PNNQ) in applications with 1D sensor data [4] and 2D moving objects [3]. In contrast, GNN query specifies multiple query points in arbitrary d-dimensional space. Only in a special case where GNN query specifies one query point, our PGNN query degrades to PNNQ. In particular, a PNNQ returns the expected probability of each object that can be the NN of a given query point. The proposed solution [3], [4] to solve the PNNQ problem includes four phases. The first *projection phase* computes the uncertainty region of each object by the application model. Then, in the second *pruning phase*, we sequentially scan the database and obtain the minimum and maximum possible distances from the query point to each uncertain object. Let $best\_dist$ be the smallest maximum distance from query point to uncertain objects. Then, all the data objects that have their minimum distances to the query point greater than or equal to $best\_dist$ can be safely pruned. Next, in the *bounding phase*, a bounding circle centered at the query point with radius $best\_dist$ can be conceptually drawn. Any data outside the bounding circle can be ignored during query processing. Finally, in the *evaluation phase*, for each remaining candidate, we calculate its expected probability of being NN of the query point, by only considering those data objects within the bounding circle. In contrast to PNNQ, our PGNN problem in the context of uncertain databases is more complex, since multiple query points are involved, and moreover, different aggregate distance functions such as sum, min, and max are used. Like the statement in [17] and [18] that the solutions for the NN query cannot be used directly for the GNN query in the "certain" database, previous methods to answer PNNQ cannot be directly applied to solve our PGNN problem in the uncertain database.

Ljosa and Singh [15] propose another equally natural interpretation of NN over uncertain data, considering the *expected distance* from query point to uncertain objects under L_1-norm. Specifically, the curve of expected distance for every possible position of query point along each dimension is offline precomputed and approximated by piecewise-linear method. Then, a tree index, APLA-tree, is constructed for range and kNN queries. In contrast, our work is more related to the natural definition of PNNQ [3] (i.e., applying the probability integration) with multiple query points, where APLA method with the expected distances cannot be used in our problem. Moreover, in this paper, we focus our work on 1-PGNN problem that retrieves the probabilistic group 1-nearest neighbors, which has not been addressed before. The variant of the k-PGNN case (obtaining probabilistic group k-nearest neighbors) will be discussed in Section 7.

3 Problem Definition

In this section, we formally define the problem of PGNN in an uncertain database $\mathcal{D}$. Specifically, assume that each data object o in $\mathcal{D}$ can be represented by an uncertainty region $UR(o)$ [3], [24], in which o locates at position $o_0 \in UR(o)$ with probability $pdf(o_0) \geq 0$ (note that, in case o_0 is not in $UR(o)$, $pdf(o_0) = 0$), where $pdf(\cdot)$ is the *probability density function* (pdf) of object o. This probabilistic representation of uncertain objects has many practical applications. For example, in sensor networks, data collected from sensors contain noises, however, often within a tolerance range; in mobile applications, the position of each mobile user can be inferred from the last reported position and the maximum moving speed. In this work, we assume that uncertain objects in the database are independent of each other. Similar assumptions have been made in some previous works [4], [3], [19].

Definition 3.1 (PGNN). *Given an uncertain database $\mathcal{D}$, a set of n query objects, $Q = \{q_1, q_2, \ldots, q_n\}$, and a user-specified probability threshold $\alpha \in (0, 1]$, a PGNN query retrieves a set of data objects $o \in \mathcal{D}$, such that they are expected to be the GNN of query set Q with a probability greater than α, that is,*

Symbols	Descriptions
$\mathcal{D}$	the data set with data size $\lvert\mathcal{D}\rvert$
Q	the user-specified query set
d	the dimensionality of the data set
n	the number of query points in Q
$UR(o)$	the *uncertainty region* of object o
α	the user-specified probability threshold
$dist(\cdot,\cdot)$	the *Euclidean distance* between two objects
$adist(o,Q)$	the *aggregate distance* from object o to query set Q
$LB\_adist(o,Q)$ ($UB\_adist(o,Q)$)	the lower (upper) bound of *aggregate distance* $adist(o,Q)$

Fig. 2. Meanings of notations.

$$\int_{r_{min}}^{r_{max}}\left(Pr\{adist(o,Q)=r\}\cdot\prod_{\forall p\in\mathcal{D}\setminus\{o\}}Pr\{adist(p,Q)\geq r\}\right)dr>\alpha, \tag{1}$$

where $adist(o,Q)$ (or $adist(p,Q)$) is defined as a monotonically increasing function

$$f(dist(o,q_1),dist(o,q_2),\ldots,dist(o,q_n))$$

(or $f(dist(p,q_1),dist(p,q_2),\ldots,dist(p,q_n))$), and $r_{min}(r_{max})$ is the minimum (maximum) possible aggregate distance $adist(o,Q)$ from o to Q.

In this paper, due to the popularity in many applications, sum, min, and max aggregate distance functions $adist(o,Q)$ (i.e., $f=sum$, min, or max) are used as presentation examples, which are denoted as $adist_{sum}$, $adist_{min}$, and $adist_{max}$, respectively. Our proposed methods, however, can be easily extended to other aggregate distance functions f, as will be discussed later. Specifically, we have

$$adist_{sum}(o,Q)=\sum_{i=1}^{n}dist(o,q_i), \tag{2}$$

$$adist_{min}(o,Q)=min_{i=1}^{n}dist(o,q_i), \tag{3}$$

$$adist_{max}(o,Q)=max_{i=1}^{n}dist(o,q_i), \tag{4}$$

where $dist(x,y)$ is the *euclidean distance* between two data objects x and y.

Intuitively, the LHS of (1) indicates the expected probability that object o is a GNN of Q. Specifically, within the integration, the first term is the probability that o has r (aggregate) distance to query set Q, whereas the second term corresponds to the probability that other data objects $p\in\mathcal{D}\setminus\{o\}$ have (aggregate) distances to Q never smaller than r (here, we can multiply probabilities of all objects due to the object independence as assumed by many previous works [4], [3], [19]). Thus, if the resulting integration for object o on the LHS of (1) is greater than a user-specified threshold $\alpha\in(0,1]$, then o is a qualified PGNN.

To the best of our knowledge, no previous work has studied the PGNN problem. Therefore, the only straightforward method is the *linear scan*, which sequentially scans all the data objects on disk and check, for each object o, whether or not it is a qualified PGNN by (1). Since (1) involves complex integration, this method requires intensive computations as well as large I/O cost, which is quite inefficient. Motivated by this, in the sequel, we propose effective filtering methods, namely, *spatial pruning* and *probabilistic pruning*, to facilitate the PGNN search, which can be integrated into our efficient PGNN query procedure. The basic idea is as follows: From the PGNN definition, our goal is to prune those data objects that have the expected probability (LHS of (1)) smaller than or equal to α, which, obviously, would not introduce any *false dismissals* (answers to the query that are, however, not in the final result). Fig. 2 summarizes the commonly used symbols in this paper.

```
Procedure PGNN_Framework {
  Input: a $d$-dimensional uncertain database $\mathcal{D}$, a set, $Q$, of $n$ query objects, $q_1$, $q_2$,
         ..., and $q_n$, and a probabilistic threshold $\alpha$
  Output: the answer set PGNN($Q$, $\alpha$)
// index construction phase
  (1)  construct a multidimensional index structure $\mathcal{I}$ over $\mathcal{D}$
// filtering phase (including spatial and probabilistic pruning)
  (2)  perform the spatial pruning over $\mathcal{I}$
  (3)  perform the probabilistic pruning over the remaining candidates
// refinement phase
  (4)  refine candidates by Inequality (1) and return the final result PGNN($q$, $\alpha$)
}
```

Fig. 3. The framework for answering PGNN queries.

4 Probabilistic GNN

Fig. 3 illustrates our general framework for retrieving PGNNs. In particular, the framework consists of three phases: index construction, filtering, and refinement. In the first phase, in order to facilitate efficient PGNN queries, we construct a multidimensional index $\mathcal{I}$ over all the uncertain objects in the database $\mathcal{D}$ (line 1). Without loss of generality, in this paper, we use one of the most popular indexes, R-tree [9], to index uncertain objects. That is, we tightly bound the uncertainty region of each object with a bounding hyperrectangle, and then insert the hyperrectangle into an R-tree, on which the PGNN query can be answered. Note that, here, we use a standard insertion operator in the R-tree. Other indexes such as M-tree [6] or SR-tree [12] can be applied as well, as our proposed methodology is independent of the underlying index. As a second step, given a set of query points, $Q=\{q_1,q_2,\ldots,q_n\}$, specified by the PGNN query, our goal is to retrieve all PGNNs of Q. Since the aggregate distance from any uncertain object to the query set is a variable instead of a fixed value, previous pruning methods [17] that handle "certain" data cannot be directly applied. Thus, in the filtering phase, we use novel pruning methods to filter out those unqualified data objects. Specifically, we propose two effective methods, namely *spatial pruning* and *probabilistic pruning*, to significantly reduce the PGNN search space yet without introducing any false dismissals (lines 2-3). Finally, in the last *refinement phase*, we further refine the remaining candidates by checking (1) and return the qualified PGNNs (line 4).

In the sequel, we focus on illustrating the PGNN pruning heuristics in the filtering phase, including both *spatial* and *probabilistic pruning*, which will be discussed in Sections 4.1 and 4.2, respectively. In particular, the *spatial pruning* approach filters out those data objects that are definitely not PGNNs, which can be applied to the case where the distribution of each object within its *uncertainty region* is

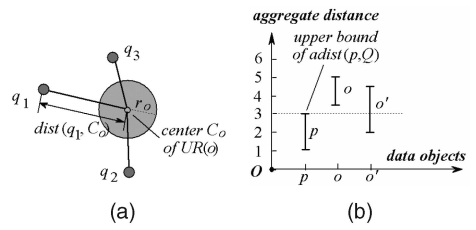


Fig. 4. Heuristics of spatial pruning.

either known or unknown. On the other hand, the *probabilistic pruning* technique corresponds to the case under the assumption that we know the distribution of each object in advance. Thus, the proposed pruning method can utilize the distribution information to achieve even higher pruning ability, that is, discarding those objects with the expected probability of being PGNNs smaller than or equal to β, for some $\beta \in [0, \alpha]$. As will be described later, both *spatial* and *probabilistic pruning* can be seamlessly integrated into our PGNN query procedure.

4.1 Spatial Pruning

First, we propose the *spatial pruning* method. Specifically, we prune those uncertain objects in the database that are definitely not PGNNs. In other words, we want to filter out those objects o with 0 expected probability (LHS of (1).

Fig. 4 illustrates the intuition of our spatial pruning method. In particular, Fig. 4a shows the uncertainty region (circle) $UR(o)$ of object o, centered at point C_o and with radius r_o, as well as a set, Q, of three PGNN query points q_1, q_2, and q_3. Note that, although object o has "dynamic" attributes, it can only locate within its uncertainty region $UR(o)$. Therefore, the distance from query point q_1 to uncertain object o is lower bounded by $(dist(q_1, C_o) - r_o)$ and upper bounded by $(dist(q_1, C_o) + r_o)$. Similarly, the distance from query point q_2 (q_3) to object o is within

$$[dist(q_2, C_o) - r_o, dist(q_2, C_o) + r_o]$$
$$([dist(q_3, C_o) - r_o, dist(q_3, C_o) + r_o]).$$

As a result, the aggregate distance $adist(o, Q)$ from o to Q can be also bounded by an interval

$$[LB\_adist(o, Q), UB\_adist(o, Q)],$$

where $LB\_adist(o, Q)$ and $UB\_adist(o, Q)$ for different aggregates (i.e., sum, min, and max) are given below.

When $adist(o, Q)$ is the sum aggregate distance function $adist_{sum}(o, Q)$, we have

$$LB\_adist_{sum}(o, Q) = \sum_{i=1}^{n} dist(C_o, q_i) - n \cdot r_o, \quad (5)$$

$$UB\_adist_{sum}(o, Q) = \sum_{i=1}^{n} dist(C_o, q_i) + n \cdot r_o. \quad (6)$$

Similarly, for the min and max aggregate distance functions, $adist_{min}(o, Q)$ and $adist_{max}(o, Q)$, respectively, we have

$$LB\_adist_{min}(o, Q) = min_{i=1}^{n} dist(C_o, q_i) - r_o, \quad (7)$$

$$UB\_adist_{min}(o, Q) = min_{i=1}^{n} dist(C_o, q_i) + r_o, \quad (8)$$

$$LB\_adist_{max}(o, Q) = max_{i=1}^{n} dist(C_o, q_i) - r_o, \quad (9)$$

$$UB\_adist_{max}(o, Q) = max_{i=1}^{n} dist(C_o, q_i) + r_o. \quad (10)$$

Fig. 4b illustrates intervals of aggregate distances for three uncertain objects p, o, and o' in the 2D *object-distance* space. In particular, assume that the aggregate distance $adist(p, Q)$ from object p to Q is within interval [1, 3]. Similarly, we have $adist(o, Q) \in [3.5, 5]$ and $adist(o', Q) \in [2, 4.5]$. Without loss of generality, suppose we have accessed uncertain object p so far and obtained its interval [1, 3]. Next, we encounter the second object o and need to decide whether or not o is a PGNN candidate. Since the lower bound distance $LB\_adist(o, Q)$ of object o (i.e., 3.5) is greater than the upper bound distance $UB\_adist(p, Q)$ of object p (i.e., 3), o is guaranteed not to be PGNN. The reason is that, within the integration of LHS of (1), it always holds that $Pr\{adist(p, Q) \geq r\} = 0$, for $r \leq UB\_adist(o, Q)$ and $adist(o, Q) \geq LB\_adist(o, Q)$. Thus, for any positive threshold α, (1) cannot hold and we can safely prune object o, which is the basic heuristics of our *spatial pruning* method. On the other hand, for the third data object o' we encounter, since it holds that

$$LB\_adist(o', Q) < UB\_adist(p, Q),$$

there exists possibility that o' becomes GNN among the three objects (i.e., p, o, and o'). So, object o' cannot be pruned and it is a PGNN candidate.

Therefore, our *spatial pruning* method can be briefly summarized as follows: Assume that object p has the smallest upper bound $UB\_adist(p, Q)$ of aggregate distance among all data objects that we have seen so far. For any data object o, as long as it holds that $LB\_adist(o, Q) \geq UB\_adist(p, Q)$, object o can be safely pruned, where $LB\_adist(o, Q)$ is the lower bound of aggregate distance from o to Q. The following lemma guarantees that *spatial pruning* method can correctly prune data objects and provide PGNN candidates.

Lemma 4.1. *No false dismissals are introduced by the spatial pruning method, while answering a PGNN query.*

Proof. By contradiction. Assume that (at least) one *false dismissal* o is introduced due to the *spatial pruning*. That is, as mentioned above, data object o satisfying $LB\_adist(o, Q) \geq UB\_adist(p, Q)$ is the answer to the PGNN query. Therefore, according to GNN definition, there must exist a position (instance) o_0 for object o and a position p_0 for object p, such that $adist(o_0, Q) < adist(p_0, Q)$ holds. However, since $LB\_adist(o, Q) \leq adist(o_0, Q)$ and $adist(p_0, Q) \leq UB\_adist(p, Q)$, by the inequality transition, we have

$$LB\_adist(o, Q) < UB\_adist(p, Q),$$

which is contrary to our initial assumption. Hence, no *false dismissals* are introduced by the *spatial pruning* method. □

Note that, in the 1D space and with one query point, the pruning criterion of our spatial pruning degrades to the one proposed in [4].

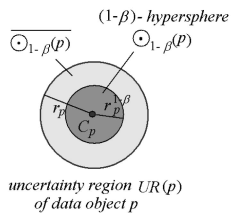


Fig. 5. Illustration of $(1-\beta)$-hypersphere.

4.2 Probabilistic Pruning

As a second step, we present a *probabilistic pruning* method to answer PGNN queries. Recall that, the *spatial pruning* method reduces the PGNN search space by pruning those data objects that are definitely not PGNNs. This method only utilizes the geometric property of uncertain objects (i.e., uncertainty regions); thus, we can apply *spatial pruning* to the case where the distribution of each object (within its uncertainty region) is unknown. Obviously, it can also be applied to the case where we know the object distribution. However, if we indeed have additional distribution information, we should be able to achieve even higher pruning power, compared with the *spatial pruning*, which motivates us to propose the *probabilistic pruning* approach.

In the sequel, we propose a novel *probabilistic pruning* approach, which utilizes the precomputed information from object distributions in their uncertainty regions. While the *spatial pruning* method discards those data objects with the expected probability (LHS of (1)) equal to zero, the *probabilistic pruning* aims to prune those objects with probability never greater than β, for some $\beta \in [0, \alpha]$, where $\alpha \in (0, 1]$ is a probability threshold specified by PGNN queries. This also indicates that *probabilistic pruning* has at least no worse pruning power than *spatial pruning*. Specifically, since the formula of the expected probability (LHS of (1)), denoted as $P_{PGNN}(o)$, is very complex involving integrations, we relax it with its upper bound probability. That is, if the upper bound probability of $P_{PGNN}(o)$ is smaller than or equal to $\beta \in [0, \alpha]$, then object o can be safely pruned. Obviously, no *false dismissals* are introduced while using this upper bound probability.

We first introduce the concept of

$$(1-\beta)\text{-hypersphere } \odot_{1-\beta}(p)$$

for an uncertain object p, which will be used in the *probabilistic pruning*.

Definition 4.1 ($(1-\beta)$-Hypersphere**).** *As illustrated in Fig. 5, given the uncertainty region $UR(p)$ of object p centered at C_p and with radius r_p, a $(1-\beta)$-hypersphere $\odot_{1-\beta}(p)$ is a hypersphere, with the same center C_p and a radius $r_p^{1-\beta}$, such that o locates in hypersphere $\odot_{1-\beta}(p)$ with probability $(1-\beta)$. The complement part of $\odot_{1-\beta}(p)$ in $UR(p)$ is denoted as $\overline{\odot_{1-\beta}(p)}$ (i.e., $\overline{\odot_{1-\beta}(p)} = UR(p) \setminus \odot_{1-\beta}(p)$).*

From the definition of $(1-\beta)$-hypersphere, we can see that the uncertainty region $UR(p)$ of object p is exactly 1-hypersphere when $\beta = 0$. Furthermore, we have an immediate corollary below.

Corollary 4.1. *Any object p locates in the complement part $\overline{\odot_{1-\beta}(p)}$ with probability β.*

Definition 4.1 assumes the continuous pdf within the uncertainty region of each object. Usually, we use discrete samples [3], [13], [14] to represent the distribution with continuous pdf. Thus, in such discrete case, we can compute $(1-\beta)$-hypersphere as follows: Assume each uncertain object has, for example, l, random samples. According to Definition 4.1, any hypersphere that contains $\lceil (1-\beta) \cdot l \rceil$ samples is a $(1-\beta)$-hypersphere of this object. (Note: In fact, there may exist many of such hyperspheres with different radii. In order to achieve the highest pruning power, however, we use the one with the smallest radius.) Thus, for each object p, we can select the geometric centroid C_p of all the samples [10], compute the distances from C_p to these samples, and finally sort samples in ascending order of distances. The radius r_p of $(1-\beta)$-hypersphere is given by the $\lceil (1-\beta) \cdot l \rceil \text{th}$ smallest distance among samples. Therefore, for each β value, the time complexity of offline computing $(1-\beta)$-hyperspheres of data set $\mathcal{D}$ is $O(|\mathcal{D}| \cdot l \cdot logl)$, where $|\mathcal{D}|$ is the data size of $\mathcal{D}$. The PGNN query with discrete pdf (i.e., with instances in each uncertain object) is similar to the continuous case (with samples). Note that, since we can obtain $(1-\beta)$-hyperspheres from samples/instances, the object distribution within the uncertainty region can be arbitrary (e.g., uniform, normal, or binomial).

Now, we give our *probabilistic pruning* method as follows: Given an uncertain database $\mathcal{D}$ and a set, Q, of PGNN query points, the *probabilistic pruning* method can prune data object o (with its aggregate distance $adist(o, Q)$ bounded by interval $[LB\_adist(o, Q), UB\_adist(o, Q)]$), if there exists an uncertain object p, such that

$$UB\_adist(p_{1-\beta}, Q) < LB\_adist(o, Q),$$

where object $p_{1-\beta}$ locates within the region of $(1-\beta)$-hypersphere for some $\beta \in [0, \alpha]$, and $UB\_adist(p_{1-\beta}, Q)$ is the upper bound of aggregate distance from $p_{1-\beta}$ to Q.

Lemma 4.2. *The probabilistic pruning method will not introduce false dismissals while answering a PGNN query.*

Proof. It is sufficient to prove, that as long as there exists an object p satisfying $UB\_adist(p_{1-\beta}, Q) < LB\_adist(o, Q)$, the expected probability $P_{PGNN}(o)$ that object o is a PGNN (i.e., LHS of (1)) is smaller than or equal to $\beta \in [0, \alpha]$. We start from the PGNN definition of object o, that is,

$$P_{PGNN}(o) = \int_{r_{min}}^{r_{max}} \Bigg(Pr\{adist(o, Q) = r\} \cdot \prod_{\forall p' \in \mathcal{D} \setminus \{o\}} Pr\{adist(p', Q) \geq r\} \Bigg) dr, \quad (11)$$

where $r_{min} = LB\_adist(o, Q)$ and $r_{max} = UB\_adist(o, Q)$.

Since each probability in (11) is within [0, 1], we have

$$P_{PGNN}(o) \leq \int_{r_{min}}^{r_{max}} \left(Pr\{adist(o,Q) = r\} \cdot Pr\{adist(p,Q) \geq r\} \right) dr, \tag{12}$$

for object $p \in \mathcal{D} \setminus \{o\}$.

Furthermore, since object p may locate in either $\odot_{1-\beta}(p)$ or $\overline{\odot_{1-\beta}(p)}$, it holds that

$$\begin{aligned} & Pr\{adist(p,Q) \geq r\} \\ = \ & Pr\{p \in \odot_{1-\beta}(p)\} \cdot Pr\{adist(p_{1-\beta},Q) \geq r | p \in \odot_{1-\beta}(p)\} \\ + \ & Pr\{p \in \overline{\odot_{1-\beta}(p)}\} \cdot Pr\{adist(\overline{p_{1-\beta}},Q) \geq r | p \in \overline{\odot_{1-\beta}(p)}\}, \end{aligned} \tag{13}$$

where $p_{1-\beta}$ and $\overline{p_{1-\beta}}$ locate in $\odot_{1-\beta}(p)$ and $\overline{\odot_{1-\beta}(p)}$, respectively.

From the assumption of the lemma, since object p satisfies the condition that

$$\begin{aligned} adist(p_{1-\beta},Q) &\leq UB\_adist(p_{1-\beta},Q) \\ &< LB\_adist(o,Q) = r_{min} \leq r, \end{aligned}$$

in the first term of (13), $Pr\{adist(p_{1-\beta},Q) \geq r | p \in \odot_{1-\beta}(p)\} = 0$ holds. Moreover, based on Corollary 4.1, we have $Pr\{p \in \overline{\odot_{1-\beta}(p)}\} = \beta$, and the second term of (13) is smaller than or equal to β (since $Pr\{adist(\overline{p_{1-\beta}},Q) \geq r | p \in \overline{\odot_{1-\beta}(p)}\} \leq 1$). Thus, from (13), we obtain $Pr\{adist(p,Q) \geq r\} \leq \beta$, which can be substituted into (12), resulting in

$$P_{PGNN}(o) \leq \beta \cdot \int_{r_{min}}^{r_{max}} (Pr\{adist(o,Q) = r\}) dr = \beta. \tag{14}$$

Hence, if object p satisfies

$$UB\_adist(p_{1-\beta},Q) < LB\_adist(o,Q),$$

it holds that $P_{PGNN} \leq \beta \leq \alpha$, indicating that object o can be safely pruned. □

From Lemma 4.2, we can precompute the radius of $(1-\beta)$-hypersphere for each uncertain object in the database, for a number of β values within [0, 1]. In order to check whether or not an object o is a qualified PGNN, we only need to test if the condition $UB\_adist(p_{1-\beta},Q) < LB\_adist(o,Q)$ holds for other objects p, using the largest precomputed β value satisfying $\beta \in [0,\alpha]$. Since the resulting precomputation for each β value takes $O(|\mathcal{D}|)$ space for data set $\mathcal{D}$, the number of β values that we can offline precompute depends on the available disk space. Moreover, the β values that we use for offline precomputation can be uniformly distributed in [0, 1], in case α specified by queries can be arbitrary. If we know the α distribution of queries (from historical data), however, we can do some optimizations by precomputing $(1-\beta)$-hyperspheres with β following the same distribution as α. This way, during query processing, the selected largest $\beta \in [0,\alpha]$ is expected to approach α.

Note that, our probabilistic pruning essentially applies the idea similar to spatial pruning. That is, it utilizes smaller offline precomputed $(1-\beta)$-hyperspheres to prune other objects, whereas the spatial pruning uses the entire uncertainty region. However, the probabilistic pruning can achieve higher pruning power (due to the smaller radius in $UB\_adist(p_{1-\beta},Q)$) and, thus, result in higher online query efficiency.

5 Query Processing

In this section, we illustrate the PGNN query processing in detail. In particular, we utilize our two pruning methods, *spatial pruning* and *probabilistic pruning*, proposed in the last section, and integrate them into the PGNN query procedure to reduce the search PGNN space. In the sequel, Section 5.1 first illustrates the pruning heuristics of pruning intermediate entries of the index. Then, Section 5.2 presents the PGNN query procedure, corresponding to filtering and refinement steps in our PGNN framework.

5.1 Pruning Intermediate Entries

As mentioned earlier, we index the uncertainty region of each data object in the R-tree, on which the PGNN query can be processed. R-tree is one of the popular multidimensional tree indexes, which recursively groups data objects with MBRs until one final node (root) is obtained. Now, we study the condition of pruning an intermediate entry in the R-tree.

Recall that, during the point-by-point pruning of the PGNN search (discussed in Section 4.1), given the smallest upper bound $UB\_adist(p,Q)$ of aggregate distance from p to Q among all the objects we have seen so far, any object $o \in \mathcal{D}$ can be safely pruned, if $UB\_adist(p,Q) \leq LB\_adist(o,Q)$ holds, where Q is a set of n query points specified by the PGNN query. Similarly, in the case of an intermediate entry e containing many uncertain objects, as long as any object x in entry e satisfies the condition that $UB\_adist(p,Q) \leq LB\_adist(x,Q)$, we can safely prune the entire entry e. However, since the exact positions of objects x in entry e are unknown without accessing its corresponding subtree, we relax the pruning condition as follows:

The Condition to Prune Intermediate Entries. *Any intermediate entry e in the R-tree can be safely pruned, if $UB\_adist(p,Q) \leq LB\_adist(e,Q)$ holds, where object p has the minimum $UB\_adist(p,Q)$ among all the objects that we have seen so far, and $LB\_adist(e,Q)$ is the minimum possible aggregate distance from any point $x \in e$ to query set Q.*

In particular, for sum, min, and max aggregate distance functions, we let

$$\begin{aligned} LB\_adist_{sum}(e,Q) &= \sum_{i=1}^{n} mindist(q_i,e), \\ LB\_adist_{min}(e,Q) &= min_{i=1}^{n} mindist(q_i,e), \\ LB\_adist_{max}(e,Q) &= max_{i=1}^{n} mindist(q_i,e), \end{aligned} \tag{15}$$

where $mindist(q_i,e)$ is the minimum distance from object q_i to any point in e.

Since (15) needs $O(n)$ computation cost for large n value, we can bound all the query points with one MBR, $\mathrm{MBR}(Q)$, and compute $LB\_adist'(e,Q)$ using $\mathrm{MBR}(Q)$ with only $O(1)$ cost. We name this pruning method the *probabilistic minimum bounding method* ($PMBM$). Specifically, for different aggregate distance functions, we have

```
Procedure PGNN_Processing {
  Input: R-tree index $\mathcal{I}$ built over $\mathcal{D}$, a query set $Q$, and a probability threshold $\alpha$
  Output: a set, $S$, of PGNNs of $Q$
  (1) initialize a min-heap $\mathcal{H}$ accepting entries in the form $(e, key)$
  (2) $S = \phi; best\_adist = +\infty$;
  (3) insert $(root(\mathcal{I}), 0)$ into heap $\mathcal{H}$
  (4) while $\mathcal{H}$ is not empty
  (5)   $(e, key)$ = de-heap $\mathcal{H}$
  (6)   if $key \geq best\_adist$, then terminate; // spatial pruning
  (7)   if $e$ is a leaf node
  (8)     for each uncertain object $o$ in $e$
  (9)       if $LB\_adist(o, Q) \leq best\_adist$ // spatial pruning
  (10)        add object $o$ to $S$
  (11)        $best\_adist = min\{best\_adist, UB\_adist(o, Q)\}$
  (12)        if we know object distributions, Probabilistic_Pruning($o$, $Q$, $\alpha$, $S$)
  (13)        else use $o$ to spatially prune candidates in $S$
  (14)  else // non-leaf node
  (15)    for each entry $e_i$ in $e$
  (16)      if $\underline{LB\_adist'(e_i, Q) \leq best\_adist}$ // $\underline{\text{Eq. (16)}}$, spatial pruning
  (17)        if $LB\_adist(e_i, Q) \leq best\_adist$ // Eq. (15), spatial pruning
  (18)          insert $(e_i, LB\_adist(e_i, Q))$ into heap $\mathcal{H}$
  (19) refine candidates in $S$ by computing the expected probability in Inequality (1)
  (20) return $S$
}
```

Fig. 6. PGNN query processing $(PMBM)$.

$$
\begin{aligned}
LB\_adist'_{sum}(e, Q) &= n \cdot mindist(MBR(Q), e),\\
LB\_adist'_{min}(e, Q) &= mindist(MBR(Q), e),\\
LB\_adist'_{max}(e, Q) &= mindist(MBR(Q), e),
\end{aligned} \tag{16}
$$

where $mindist(MBR(Q), e)$ is the minimum possible distance between two points coming from $MBR(Q)$ and e, respectively.

From (16), the performance (e.g., pruning power) of $PMBM$ depends on the property of query set Q. For example, when query points in Q are very sparse, the resulting $MBR(Q)$ would be large; thus, $LB\_adist(e, Q)$ in (16) would be very small, leading to low pruning power. Thus, instead of bounding all the query points q_i (for $i \in [1, n]$) with an MBR as $PMBM$ does, we also propose a *probabilistic single point method* $(PSPM)$. Specifically, we use the geometric centroid q of all the query points as representative to prune intermediate nodes. That is, by applying the *triangle inequality*, we can obtain $LB\_adist(e, Q)$ with $O(1)$ cost by letting

$$
\begin{aligned}
LB\_adist''_{sum}(e, Q) &= n \cdot mindist(q, e) - \sum_{i=1}^{n} dist(q_i, q),\\
LB\_adist''_{min}(e, Q) &= mindist(q, e) - max_{i=1}^{n} dist(q_i, q),\\
LB\_adist''_{max}(e, Q) &= mindist(q, e) - min_{i=1}^{n} dist(q_i, q),
\end{aligned} \tag{17}
$$

where $dist(q_i, q)$ $(1 \leq i \leq n)$ are calculated only once when query set Q arrives.

5.2 PGNN Query Procedure

Up to now, we have discussed the pruning heuristics of intermediate nodes in the R-tree index $\mathcal{I}$, as well as the point-by-point pruning. Next, we present the detailed PGNN query procedure over $\mathcal{I}$ (using $PMBM$ to prune intermediate entries) in Fig. 6. Specifically, procedure PGNN_Processing takes a set, Q, of query points and a probability threshold α as input, and returns a PGNN set, S, by traversing the R-tree index $\mathcal{I}$ in a *best-first* manner.

In particular, we maintain a minimum heap $\mathcal{H}$ accepting entries in the form (e, key) (line 1), where e is the node MBR in the R-tree, and key is the sorting key of the heap which is defined as the lower bound of aggregate distance from any point in MBR node to query set Q. Moreover, we also initialize a PGNN candidate set S to be empty and set variable $best\_adist$ to $+\infty$ (line 2), where $best\_adist$ is the smallest upper bound of aggregate distance for all the data objects that have been seen so far. Then, we insert the root, $root(\mathcal{I})$, of the R-tree into heap $\mathcal{H}$ (line 3). Each time we pop out an entry (e, key) from heap $\mathcal{H}$ (line 5), and check whether or not key is greater than or equal to $best\_adist$. If the answer is yes, then it indicates that all the remaining entries in heap $\mathcal{H}$ would have their minimum aggregate distance to Q not smaller than $best\_adist$. Thus, they cannot contain any PGNNs and procedure PGNN_Processing can terminate (line 6).

During the PGNN search, whenever we encounter a leaf node e, for each uncertain object o in e, we check by *spatial pruning* whether or not the lower bound distance $LB\_adist(o, Q)$ from object o to query set Q is smaller than or equal to $best\_adist$ (lines 7-9). If the *spatial pruning* method cannot prune object o, then we add o to PGNN candidate set S (line 10), and update $best\_adist$ with $UB\_adist(o, Q)$ (line 11). In case we know object distributions in their uncertainty regions, we can perform the *probabilistic pruning* over the candidate set S (line 12); otherwise, we use o to spatially prune candidates in S (line 13).

In case we encounter a nonleaf node e, for each entry e_i in e, we try to prune e_i using the lower bound $LB\_adist(e_i, Q)$ of aggregate distance from entry e_i to query set Q. Specifically, we first prune entry e_i by utilizing the lower bound $LB\_adist'(e_i, Q)$ given in (16) (line 16), which requires only $O(1)$ computation cost. Note that, here we use $PMBM$ to prune intermediate nodes by MBR(Q). For $PSPM$, however, we can easily replace the underlined part of line 16 with the lower bound $LB\_adist''(e_i, Q)$ given in (17). If entry e_i cannot be pruned (i.e., either $LB\_adist'(e_i, Q) \leq best\_adist$ holds for $PMBM$ or $LB\_adist''(e_i, Q) \leq best\_adist$ holds for $PSPM$), we apply a tighter lower bound distance $LB\_adist(e_i, Q)$ given in (15) (line 17), which has higher $O(n)$ computation cost and high pruning power as well. When both lower bounds cannot prune entry e_i, we insert it into heap $\mathcal{H}$ with the key $LB\_adist(e_i, Q))$ for the further filtering. This procedure repeats until either heap $\mathcal{H}$ is empty (line 4) or line 6 holds.

After the filtering phase with *spatial pruning* and *probabilistic pruning*, we start to refine the remaining PGNN candidates in S (i.e., the refinement phase, line 19). In particular, for each candidate o in S, we calculate the expected probability $P_{PGNN}(o)$ that o is PGNN of Q (given by LHS of (1)). Note that, here we compute the expected probability (integration) by applying the numerical method, similar to that used in [4] and [3]. If $P_{PGNN}(o)$ is smaller than or equal to the given probabilistic threshold α, then object o can be safely discarded; otherwise, it is a qualified answer to the PGNN query. One interesting observation is that, although the integration on the LHS of (1) requires to scan all the data objects in the database (i.e., compute $\prod_{\forall p \in \mathcal{D} \setminus \{o\}} Pr\{adist(p, Q) \geq r\}$ for all objects $p \in \mathcal{D} \setminus \{o\}$), we can ignore those data objects p' that are not in the candidate set S yet without introducing any inaccuracy. The reason

```
Procedure Probabilistic_Pruning {
  Input: new candidate $o$, a query set $Q$, a probability threshold $\alpha$, a candidate set $S$
  (1) for each object $p$ in $S\backslash\{o\}$
  (2)     if $p$ is probabilistically invalidated by $o$ // invalidate other candidates $p$ in $S$
  (3)         mark $p$ as false alarm
  (4) load $(1-\beta)$-hypersphere $UR_{1-\beta}(o)$ for the largest pre-computed $\beta \in [0, \alpha]$
  (5) if $o$ is probabilistically invalidated by PGNN candidates in $S\backslash\{o\}$
  (6)         mark $o$ as false alarm
}
```

Fig. 7. Procedure of probabilistic pruning.

for this is as follows: According to our query procedure, any object p' that is pruned directly by spatial pruning must have its lower bound distance higher than $best\_adist$. In other words, it always holds that $Pr\{adist(p', Q) \geq r\} = 1$, where r is the aggregate distance of a candidate in S. Thus, for objects with aggregate distance beyond $best\_adist$, 1 is multiplied into integration, which would not affect the final result. Note that, when applying the probabilistic pruning, we also need to consider those objects in S that are marked as *false alarms* (as given in procedure Probabilistic_Pruning). Although these objects are pruned by probabilistic pruning, they may still be involved in calculating the probability integration later. Since the threshold $best\_adist$ that bounds S is given by spatial pruning in procedure PGNN_Processing, the contents of S with and without probabilistic pruning are the same (the only difference is that for S with probabilistic pruning, some candidates in S are not necessary to be refined in the refinement step if they are marked as false alarms). Therefore, we can correctly compute the probability for the probabilistic pruning. Assuming the computation of one probability $Pr\{\cdot\}$ requires $O(1)$ cost, the cost of refining each candidate is, thus, $O(|S|)$ in our query procedure. Thus, the total refinement cost of our method is given by $O(|S|^2)$, which is smaller than $O(|S| \cdot |\mathcal{D}|)$ directly using (1).

Finally, we discuss procedure Probabilistic_Pruning in line 12 of procedure PGNN_Processing. Note that, in procedure PGNN_Processing, we issue the probabilistic pruning only if object o cannot be spatially pruned. This is because the probabilistic pruning requires to load the $(1-\beta)$-hypersphere of object o from the disk (line 4 of procedure Probabilistic_Pruning), which is costly (if we do that for every object we encounter in the leaf nodes). Thus, in order to reduce the cost, we load the $(1-\beta)$-hypersphere only for those PGNN candidates in the candidate set S. After we add a PGNN candidate o to S, we invoke procedure Probabilistic_Pruning to further filter out *false alarms* in the candidate set S, whose details are illustrated in Fig. 7. In particular, via *probabilistic pruning*, we first use object o to invalidate existing candidates p in $S \setminus \{o\}$ (lines 1-3) and then symmetrically use other candidates to invalidate o (lines 4-6). If any object is invalidated, we simply mark it as an unqualified PGNN candidate (line 3 or 6). For the *probabilistic pruning* approach, given a new candidate o, we invalidate existing candidate $p \in S$ by using the $(1-\beta)$-hypersphere $\odot_{1-\beta}(o)$ of object o, where β is the largest precomputed value such that $\beta \in [0, \alpha]$ (as given by Lemma 4.2); vice versa.

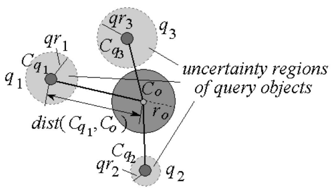


Fig. 8. Illustration of the PGNN query with uncertain query objects.

6 The PGNN Query with Uncertain Query Objects

Up to now, we have discussed the PGNN query processing when the query points are *precise*. In this section, we further consider the case where query objects of PGNN queries are also *uncertain*. Fig. 8 illustrates an example of the PGNN query with uncertain query objects. Assume we have three uncertain query objects q_1, q_2, and q_3, which correspond to their *uncertainty regions* $UR(q_1)$, $UR(q_2)$, and $UR(q_3)$ centered at C_{q_1}, C_{q_2}, and C_{q_3}, with radii qr_1, qr_2, and qr_3, respectively. Based on the PGNN definition, we want to retrieve all data objects (e.g., o) satisfying the condition in (1) (i.e., the expected probability of being GNN is greater than a user-specified probability threshold α).

Similar to the PGNN query processing with precise query points, we have two corresponding pruning methods, *spatial* and *probabilistic pruning*. For the *spatial pruning*, we only need to redefine the lower bound of the aggregate distance $LB\_adist(o, Q)$ ($LB\_adist(e, Q)$) from data object o (entry e) to query set Q. In particular, as illustrated in Fig. 8, the minimum (maximum) possible distance from query object q_1 to object o is given by

$$dist(C_{q_1}, C_o) - r_o - qr_1 \; (dist(C_{q_1}, C_o) + r_o + qr_1).$$

Similarly, for query objects q_2 and q_3, we can also obtain their minimum (maximum) distances to object o. Thus, this way, we can compute the lower (upper) bound of different aggregate distance functions $adist(o, Q)$ defined as follows:

$$\begin{aligned}
LB\_adist_{sum}(o, Q) &= \sum_{i=1}^{n} dist(C_o, q_i) - n \cdot r_o - \sum_{i=1}^{n} qr_i,\\
UB\_adist_{sum}(o, Q) &= \sum_{i=1}^{n} dist(C_o, q_i) - n \cdot r_o + \sum_{i=1}^{n} qr_i,\\
LB\_adist_{min}(o, Q) &= min_{i=1}^{n} dist(C_o, q_i) - r_o - max_{i=1}^{n} qr_i,\\
UB\_adist_{min}(o, Q) &= min_{i=1}^{n} dist(C_o, q_i) - r_o + max_{i=1}^{n} qr_i,\\
LB\_adist_{max}(o, Q) &= max_{i=1}^{n} dist(C_o, q_i) - r_o - max_{i=1}^{n} qr_i,\\
UB\_adist_{max}(o, Q) &= max_{i=1}^{n} dist(C_o, q_i) - r_o + max_{i=1}^{n} qr_i,
\end{aligned} \tag{18}$$

where qr_i is the radius of $UR(q_i)$ for uncertain query object q_i for $i \in [1, n]$.

Furthermore, during the query processing of PGNN over the R-tree index, we need to redefine the lower bound $LB\_adist(e, Q)$ of aggregate distance from an entry e to query set Q, in order to prune the PGNN search space. For the $PMBM$ method, we rewrite (16) as

$$LB\_adist'_{sum}(e,Q) = n \cdot mindist(MBR(Q'),e) - \sum_{i=1}^{n} qr_i,$$
$$LB\_adist'_{min}(e,Q) = mindist(MBR(Q'),e) - max_{i=1}^{n} qr_i,$$
$$LB\_adist'_{max}(e,Q) = mindist(MBR(Q'),e) - max_{i=1}^{n} qr_i, \quad (19)$$

where $\mathrm{MBR}(Q')$ is an MBR bounding centers C_{q_i} of $UR(q_i)$ for all $i \in [1,n]$.

Similarly, for the $PSPM$ method, we rewrite (17) as

$$\begin{aligned} & LB\_adist''_{sum}(e,Q) \\ =\ & n \cdot mindist(q,e) - \sum_{i=1}^{n} dist(q_i,q) - \sum_{i=1}^{n} qr_i, \\ & LB\_adist''_{min}(e,Q) \\ =\ & mindist(q,e) - max_{i=1}^{n} dist(q_i,q) - max_{i=1}^{n} qr_i, \\ & LB\_adist''_{max}(e,Q) \\ =\ & mindist(q,e) - min_{i=1}^{n} dist(q_i,q) - max_{i=1}^{n} qr_i, \end{aligned} \quad (20)$$

where q is the *geometric centroid* of centers C_{q_i} of $UR(q_i)$ for all $i \in [1,n]$.

Therefore, in the case of uncertain query objects, either (19) or (20) can be used in line 15 of procedure PGNN_Processing to prune entries e_i, whereas (18) can be applied to line 9 to prune data objects o.

For the *probabilistic pruning*, similarly, we need to update the lower bound distances. Specifically, given an uncertain database $\mathcal{D}$, data object $o \in \mathcal{D}$ can be safely pruned, if there exists an uncertain object p, such that $UB\_adist(p_{1-\beta},Q) < LB\_adist(o,Q)$, where the lower (upper) bound $LB\_adist(o,Q)$ $(UB\_adist(p_{1-\beta},Q))$ of aggregate distance $adist(o,Q)$ $(adist(p_{1-\beta},Q))$ are defined in (18).

With this update, we have the following lemma:

Lemma 6.1. *The probabilistic pruning with an uncertain query object set will not introduce false dismissals while answering a PGNN query.*

Proof. The proof is similar to that in Lemma 4.2. In particular, the key step of the proof is the derivation of (13). By applying the new definition of $LB\_adist(o,Q)$ and $adist(p_{1-\beta},Q)$ in (18), from the assumption of the lemma, since object p satisfies the condition that

$$adist(p_{1-\beta},Q) \le UB\_adist(p_{1-\beta},Q) < LB\_adist(o,Q) = r_{min} \le r,$$

in the first term of (13), $Pr\{adist(p_{1-\beta},Q) \ge r | p \in \odot_{1-\beta}(p)\} = 0$ holds. The other proof procedures are the same. □

Thus, this way, we can apply both *spatial* and *probabilistic pruning* to query procedure PGNN_Processing in Fig. 6, to handle the case of uncertain query objects.

7 Other Variants of PGNN Queries

In this section, we discuss some variants of PGNN queries. First, since the PGNN query is not restricted to sum, min, and max aggregates (as discussed above), it can be defined by any monotonically increasing distributive aggregate distance functions f. Note that, f is a distributive aggregate function [11], [16] if and only if there exists a function g such that $f(I') = g(f(I),v)$, where $f(I)$ is the aggregate value, $f(I')$ is the aggregate after updating with value v. Aggregates like sum, min, and max are distributive aggregates, however, $median$ is not a distributive aggregate (thus, we cannot apply our methods and have to access the entire database in order to obtain the query results).

We list a few PGNN variants with distributive aggregates as follows: For $mean$ function, given an uncertain object o and a query set Q, we have the aggregate distance function

$$adist_{mean}(o,Q) = f(dist(o,q_1), dist(o,q_2), \ldots, dist(o,q_n)) = \frac{\sum_{i=1}^{n} dist(o,q_i)}{n}.$$

Note that, since n can be considered as a constant once it is specified by the query, the PGNN query with the $mean$ aggregate would have the same PGNN results as that with sum due to $adist_{mean}(o,Q) = adist_{sum}(o,Q)/n$ for any object o in the database. For other types of aggregate distance functions f, we can compute the lower (upper) bound of $adist_f(o,Q)$ (or $adist_f(e,Q)$) from any uncertain object o (or MBR node e) to query set Q, similar to sum, min, or max, which can be used to facilitate pruning data objects/MBR nodes.

Previously, we only consider the aggregate distance function f as a function of $dist(o,q_1), dist(o,q_2), \ldots,$ and $dist(o,q_n)$ with equal weight $w(=1)$. One variant of the PGNN query can be that with a weighted aggregate distance function. For example, the weighted sum function, $adist_{f_W}(o,Q)$, can be defined as

$$adist_{f_W}(o,Q) = f_W(dist(o,q_1), dist(o,q_2), \ldots, dist(o,q_n)) = \sum_{i=1}^{n} (w_i \cdot dist(o,q_i)),$$

where w_i is a positive weight for query point q_i. In this case, our proposed method can be easily extended by redefining the lower (upper) bound of $adist_{f_W}(o,Q)$ (or $adist_{f_W}(e,Q)$) incorporating the weights w_i. For example,

$$LB\_adist_{sum_W}(o,Q) = \sum_{i=1}^{n} (w_i \cdot dist(C_o,q_i)) - r_o \cdot \sum_{i=1}^{n} w_i.$$

In this paper, we will not discuss the details.

In addition to PGNN query with different aggregate functions, we also discuss a variant of PGNN, namely, k-PGNN, which retrieves uncertain objects that are expected to be group k-nearest neighbors with probability at least $\alpha \in (0,1]$. Formally, given a database $\mathcal{D}$, we want to retrieve objects $o \in \mathcal{D}$ such that

$$\begin{aligned} P_{k-PGNN}(o) = \int_{r_{min}}^{r_{max}} & (Pr\{adist(o,Q) = r\} \\ & \cdot \sum_{\forall T \subseteq \mathcal{D}} \prod_{\forall t \in T} Pr\{adist(t,Q) \le r\} \\ & \cdot \prod_{\forall p \in \mathcal{D} \setminus (T \cup \{o\})} Pr\{adist(p,Q) \ge r\}) dr > \alpha, \end{aligned}$$

where T is a subset of $\mathcal{D}$ with size $(k-1)$. Intuitively, the answer set of the k-PGNN query contains objects o such that

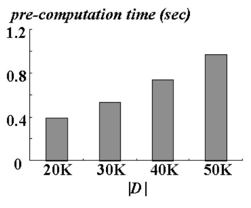


Fig. 9. Precomputation cost versus data size $|\mathcal{D}|$.

with high probability ($> \alpha$), there exist $(k-1)$ objects (e.g., that in T) whose aggregate distances are not greater than $adist(o,Q)$ and other objects have their aggregate distances not smaller than $adist(o,Q)$. Although the formula of $P_{k-PGNN}(o)$ is more complex than the PGNN definition, the pruning idea is similar. Specifically, for the spatial pruning, instead of finding minimum upper bound of aggregate distance as the pruning threshold in the PGNN problem, in the $k-$ PGNN case, we obtain the kth smallest upper bound of aggregate distance, k_best_adist, and safely prune those objects whose lower bound distances are not smaller than k_best_adist. The probabilistic pruning cannot be applied to handle the k-PGNN case, since there is a summation defined in the formula of $P_{k\text{-}PGNN}(o)$, which is different from the PGNN case.

8 Experimental Evaluation

In this section, we empirically evaluate the efficiency and effectiveness of our proposed pruning methods (including both *spatial* and *probabilistic pruning*) to answer the PGNN query. Since the real data sets are not available, we test our approaches over synthetic data sets in a d-dimensional data space, $\mathcal{U} = [0, 1,000]^d$, similar to [3], [5], and [24]. In particular, we generate uncertain objects as follows: For each uncertain object $o \in \mathcal{D}$, we first decide the center location C_o of its *uncertainty region* $UR(o)$, and then randomly produce the radius r_o of $UR(o)$ within $[r_{min}, r_{max}]$. For brevity, we denote $lUrU$ ($lUrG$) as the data set with center *locations* C_o of *Uniform* distribution and *radius* $r_o \in [r_{min}, r_{max}]$ of *Uniform* distribution (*Gaussian* distribution, with mean $(r_{min} + r_{max})/2$ and variance $(r_{max} - r_{min})/5$). Similarly, $lSrU$ ($lSrG$) represents the data set with center *locations* C_o of *Skew* ($Zipf$) distribution $(\text{skewness} = 0.8)$ and *radius* $r_o \in [r_{min}, r_{max}]$ of *Uniform* distribution (or *Gaussian* with the same settings). In case we know the object distributions within their uncertainty regions, for each uncertain object o, we can offline precompute $(1-\beta)$-hypersphere $UR_{1-\beta}(o)$ for some $\beta \in [0,1]$. Fig. 9 shows the precomputation time for each β value with different data sizes $|\mathcal{D}|$, where the dimensionality is 3, and 100 samples per object are used for probability integration. Since different object distributions may produce different sizes (i.e., radii) of $UR_{1-\beta}(o)$, the resulting bounds of aggregate distances would be different, which also indicates different pruning abilities during the probabilistic pruning (note: the spatial pruning would not be affected). For simplicity, in our experiments, we assume that each object $o \in \mathcal{D}$ is uniformly distributed within its *uncertainty region* $UR(o)$. We also did experiments over data sets with other parameter values or object distributions, however, due to space limit, we do not present the results (which have the similar trends as the results obtained from uniform distributions). After generating data sets, we index them with R-tree [9], on which PGNN queries can be processed, where the page size is set to 4 Kbytes.

In order to produce PGNN (precise) query points, we first generate a random point in the data space $\mathcal{U}$ following the same distribution as center locations of data objects. Next, within a hyperrectangle centered at the generated point and with a side length $(1{,}000 \times \Delta)$ $(\Delta \in [0,1])$ along each dimension, we randomly pick up n query points as the input of the PGNN query, where Δ is a parameter indicating the size of $MBR(Q)$ for all query points. For PGNN queries with uncertain query objects, we first generate n precise points mentioned above as n centers of uncertain query objects, respectively, and then randomly produce the radius (of *uncertainty region*) for each query object within $[qr_{min}, qr_{max}]$, where the radius distribution of uncertain query objects is the same as that of the data set (i.e., either *uniform* or *Gaussian*).

In our experiments, we use the *wall clock time* to measure the filtering time of PGNN query processing, which is the time cost of retrieving (filtering) PGNN candidates (excluding the time for refining PGNN candidates by probability integrations in (1)). It consists of two portions, CPU time and I/O cost, where we incorporate the cost of each page access (i.e., I/O) by penalizing 10 ms [25], [26].

To the best of our knowledge, no previous work has studied the PGNN problem in the context of uncertain databases. Therefore, the only available method to answer PGNN queries is the *linear scan*, which sequentially scans all the objects on disk for each PGNN candidate in order to check the condition in (1). However, this is very costly. As an example, given 30,000 3D data objects and 4 Kbytes page size, even the I/O cost of the linear scan method (somewhat like nested loop operation) requires $\frac{10\text{ ms}}{1{,}000\text{ ms/s}} \times \frac{30{,}000 \times 3 \times 4}{4{,}000} \times \frac{30{,}000 \times 3 \times 4}{4{,}000} = 81\text{ seconds}$ (assuming a floating number needs 4 bytes). In the sequel, we will list the *speed-up ratio* of our methods, defined as the *wall clock time* of *linear scan* divided by that of our methods. All our experiments are conducted on Pentium IV 3.2-GHz PC with 1-Gbyte memory, and the reported results are the average of 1,000 queries.

8.1 Performance versus β

In the first set of experiments, we evaluate the query performance of our proposed approaches, namely, $PMBM$ and $PSPM$, in terms of the *wall clock time*, in which the *probabilistic pruning* is processed under different precomputed values of β. In particular, as illustrated in Section 4.1, $PMBM$ bounds all the query points with an MBR and uses this query MBR to help prune objects/nodes. In contrast, upon the arrival of PGNN queries, $PSPM$ first computes the *geometric centroid* of (precise) query points and then uses this centroid to facilitate reducing the PGNN search space. In order to apply the *probabilistic pruning* for both $PMBM$ and $PSPM$, we precompute the radius of $(1-\beta)$-hyperspheres of objects offline, for some $\beta \in [0,1]$. Whenever a PGNN query with probability threshold $\alpha \in [0,1]$ arrives, we would use

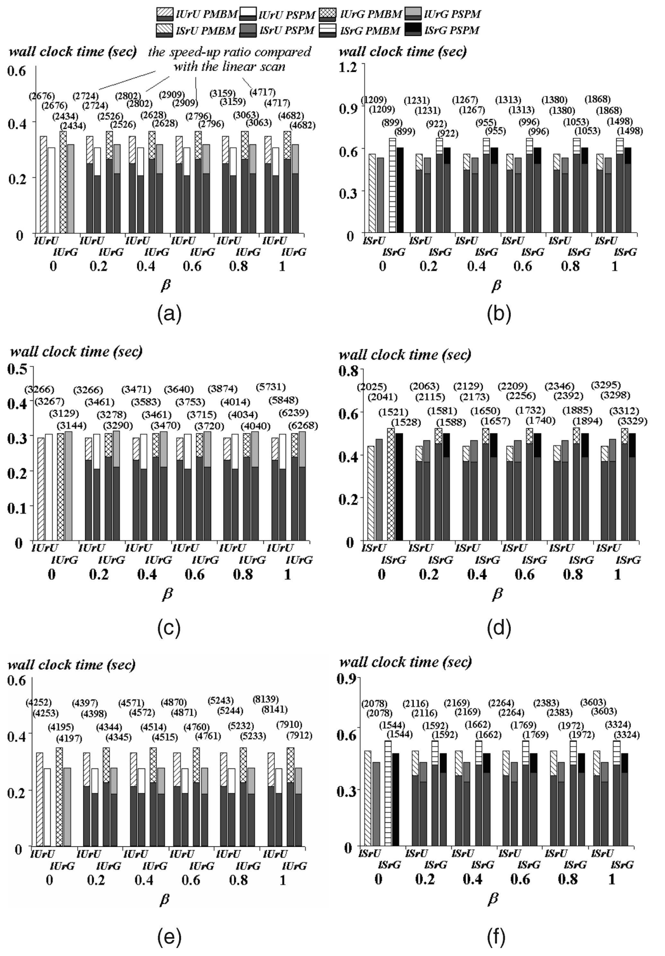


Fig. 10. Performance versus β (precise query points). (a) $lUrU$ and $lUrG$ (sum). (b) $lSrU$ and $lSrG$ (sum). (c) $lUrU$ and $lUrG$ (min). (d) $lSrU$ and $lSrG$ (min). (e) $lUrU$ and $lUrG$ (max). (f) $lSrU$ and $lSrG$ (max).

the largest precomputed β value satisfying $\beta \in [0, \alpha]$ for the *probabilistic pruning*.

Given a PGNN query with precise query points and $\alpha = 1$, Fig. 10 illustrates the query performance of our two approaches over data sets $lUrU$, $lUrG$, $lSrU$, and $lSrG$, in case the largest precomputed β value is 0, 0.2, 0.4, 0.6, 0.8, or 1, where the radius range $[r_{min}, r_{max}] = [0, 15]$, the dimensionality $d = 3$, the number of PGNN query points, $n = 4$, the parameter $\Delta = 10$ percent, and the data size $|\mathcal{D}| = 30{,}000$. During the refinement phase, we assume that 100 random samples are used for each uncertain object to compute the probability integration in (1). Note that, in all figures, each stacked column has two parts, the time cost of spatial pruning (the upper part) and that of probabilistic pruning invoked by line 12 of procedure PGNN_Processing (the lower part); the numbers in brackets over columns are the *speed-up ratio* of our methods, compared with the *linear scan*.

The experimental results in Fig. 10 show that, when β increases, the filter cost of both $PMBM$ and $PSPM$ remains approximately the same but low (i.e., less than 0.7 second), in terms of *wall clock time*, for all the four types of data sets and *aggregate distances*, sum (Figs. 10a and 10b), min (Figs. 10c and 10d), and max (Figs. 10e and 10f). When

TABLE 1
Average Number of Overlapping Objects versus $[r_{min}, r_{max}]$

AVG No. of Overlapping Objects / $[r_{min}, r_{max}]$	$lUrU$	$lUrG$	$lSrU$	$lSrG$
[0, 10]	0.178739	0.153272	2.15387	1.93593
[0, 15]	0.60742	0.50035	6.32961	5.72392
[0, 20]	1.47072	1.20004	13.3833	12.1669
[0, 25]	2.84316	2.34254	23.6475	21.5765

$\beta = 0$, our query procedure, PGNN_Processing, only uses the *spatial pruning* to prune objects and nodes; when $\beta > 0$, both spatial and probabilistic pruning are applied. From the figures, our approaches can outperform the *linear scan* by about three orders of magnitude, showing the efficiency of our query procedure. Moreover, the *speed-up ratio* becomes larger with the increasing β, which confirms the effectiveness of our *probabilistic pruning*. This is reasonable since we always prune those objects with the expected probability smaller than or equal to β. Thus, more objects are expected to be pruned with large β. Furthermore, $PMBM$ and $PSPM$ are comparable with similar speed-up ratios.

After evaluating the query performance with different β values, in the sequel, we will study the robustness of our proposed methods for PGNN query processing, over the four types of synthetic data sets, under different parameter settings. In particular, by fixing $\beta = 0.8$, we vary the values of different parameters (including $[r_{min}, r_{max}]$, d, n, Δ, $|\mathcal{D}|$, and $[qr_{min}, qr_{max}]$) and verify the effectiveness of our pruning methods as well as the efficiency of the PGNN query procedure. Due to space limit, we would only present results of sum and min aggregates (that of max is similar to min). In the sequel, we first consider PGNN queries with precise query points (i.e., $qr_{min} = qr_{max} = 0$), whereas the experimental results with uncertain query objects will be presented in the last section.

8.2 Performance versus $[r_{min}, r_{max}]$

In this section, we study the effect of the radius range $[r_{min}, r_{max}]$ of *uncertainty regions* $UR(o)$ on the PGNN query performance. Specifically, we vary the radius range $[r_{min}, r_{max}]$ to [0, 10], [0,15], [0, 20], and [0, 25], where $\beta = 0.8$, the dimensionality $d = 3$, the number of PGNN query points, $n = 4$, the parameter $\Delta = 10$ percent, the data size $|\mathcal{D}| = 30{,}000$, and the number of samples per object used for probability integration is 100. Table 1 illustrates the average number of objects that overlap with an object in different data sets of $[r_{min}, r_{max}]$. In particular, when the radius range becomes larger, the number of overlapping objects increases from about 0.2 to 3 for lU data sets and from about 2 to 20 for lS data sets.

Fig. 11 shows the experimental results on four types of data sets, with sum (Figs. 11a and 11b) and min (Figs. 11c and 11d) aggregate distance functions. When the range $[r_{min}, r_{max}]$ increases from [0, 10] to [0, 25], the required *wall clock time* of both $PMBM$ and $PSPM$ becomes higher. Correspondingly, the *speed-up ratio* of our methods smoothly decreases with the increasing radius range $[r_{min}, r_{max}]$. This is mainly because, if uncertain objects can locate anywhere in larger *uncertainty regions*, their chances of being GNN would increase. Thus, more PGNN candidates have to be included

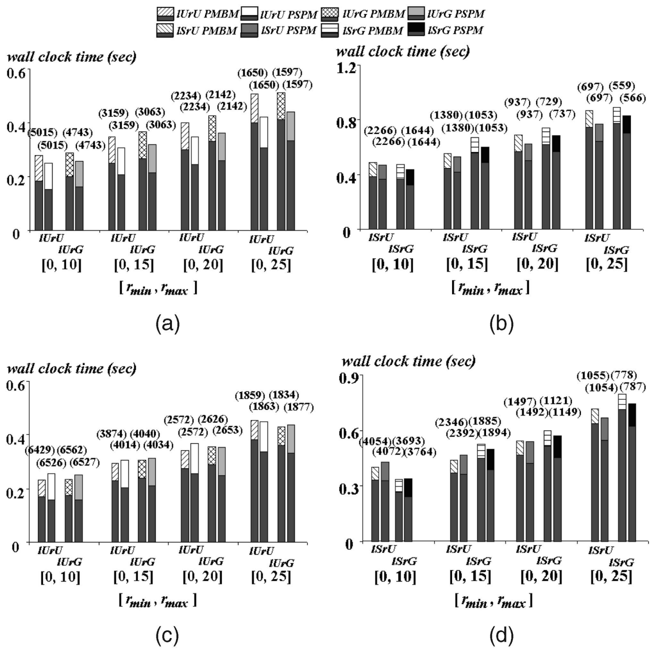


Fig. 11. Performance versus $[r_{min}, r_{max}]$ (precise query points). (a) $lUrU$ and $lUrG$ (sum). (b) $lSrU$ and $lSrG$ (sum). (c) $lUrU$ and $lUrG$ (min). (d) $lSrU$ and $lSrG$ (min).

in the candidate set, which results in higher PGNN retrieval cost. However, the *speed-up ratio* of our methods remains high (i.e., by three orders of magnitude). Similar to previous results, $PSPM$ and $PMBM$ are comparable in terms of *wall clock time* and *speed-up ratio*.

8.3 Performance versus Dimensionality d

Next, we study the effect of dimensionality on the PGNN query performance. Fig. 12 illustrates experimental results of $PMBM$ and $PSPM$ where the dimensionality $d = 2, 3, 4, 5$, $\beta = 0.8$, the radius range $[r_{min}, r_{max}] = [0, 15]$, the number of (precise) PGNN query points, $n = 4$, the parameter $\Delta = 10$ percent, the data size $|\mathcal{D}| = 30{,}000$, and the number of samples per object used for probability integration is 100. Table 2 shows the average number of overlapping objects with an object for different data sets. Since the data sizes of all the data sets are fixed to 30,000, higher dimensionality results in less overlapping. In Fig. 12, we find, that when the dimensionality, d, of data sets is 2, the *wall clock time* is the highest. This is reasonable, since 30,000 (uncertain) data objects in the 2D space are quite dense (as confirmed by Table 2), compared to those with the same data size in 3D, 4D, and 5D spaces. Therefore, many data objects are qualified as candidates in the 2D data sets, which alternatively increases the query processing cost in terms of the *wall clock time*. Compared with the *linear scan*, the *speed-up ratios* of $PMBM$ and $PSPM$ are similar and they increase with the increasing dimensionality due to fewer PGNN candidates to refine.

TABLE 2
Average Number of Overlapping Objects versus Dimensionality

AVG No. of Overlapping Objects / d	$lUrU$	$lUrG$	$lSrU$	$lSrG$
2	24.4423	22.6574	116.548	105.661
3	0.60742	0.50035	6.32961	5.72392
4	0.01467	0.01200	0.33648	0.24561
5	0.00040	0.00013	0.01740	0.01327

8.4 Performance versus Query Size n

Fig. 13 varies the query size, that is, the number, n, of (precise) query points specified by PGNN queries, and evaluates the query performance of $PMBM$ and $PSPM$. In particular, we set the query size n to 3, 4, 5, 6, where $\beta = 0.8$, the radius range $[r_{min}, r_{max}] = [0, 15]$, the dimensionality $d = 3$, the parameter $\Delta = 10$ percent, the data

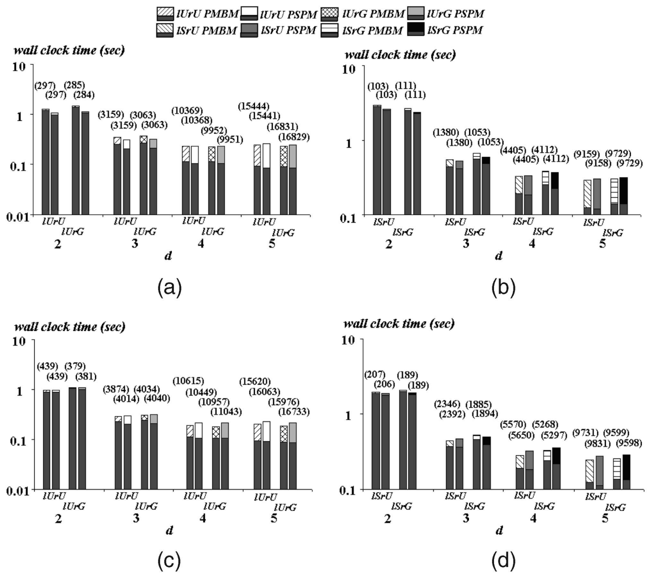


Fig. 12. Performance versus dimensionality d (precise query points). (a) $lUrU$ and $lUrG$ (sum). (b) $lSrU$ and $lSrG$ (sum). (c) $lUrU$ and $lUrG$ (min). (d) $lSrU$ and $lSrG$ (min).

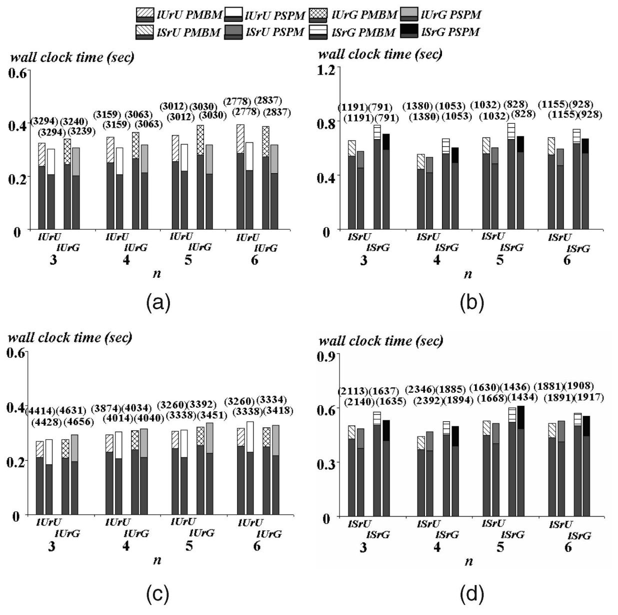


Fig. 13. Performance versus n (precise query points). (a) $lUrU$ and $lUrG$ (sum). (b) $lSrU$ and $lSrG$ (sum). (c) $lUrU$ and $lUrG$ (min). (d) $lSrU$ and $lSrG$ (min).

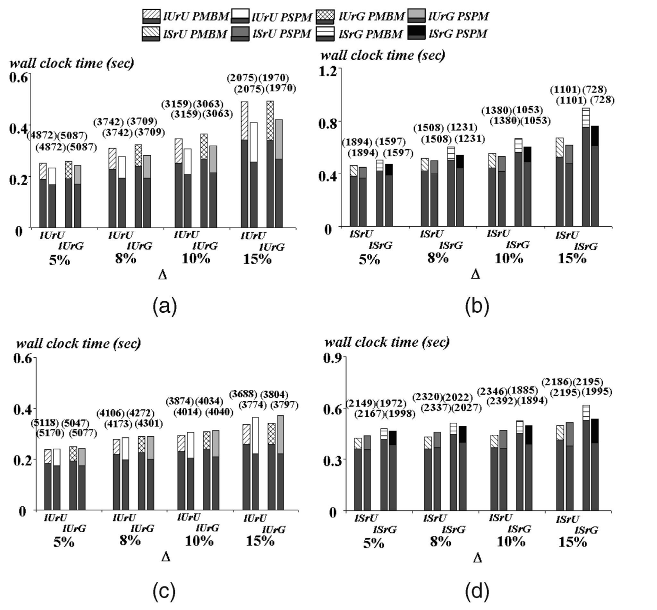


Fig. 14. Performance versus Δ (precise query points). (a) $lUrU$ and $lUrG\ (sum)$. (b) $lSrU$ and $lSrG\ (sum)$. (c) $lUrU$ and $lUrG\ (min)$. (d) $lSrU$ and $lSrG\ (min)$.

size $|\mathcal{D}| = 30{,}000$, and the number of samples per object used for probability integration is 100. In general, for lU data sets with uniform location distributions, the filtering cost is increasing with respect to the increasing n (since we need to calculate the lower bound distance from each object/node to n query points during the PGNN query processing); for lS data sets, the filtering time varies due to the skew location distribution of objects (i.e., the number of retrieved PGNN candidates varies). Moreover, both $PMBM$ and $PSPM$ can outperform the *linear scan* with the speed-up ratio by about three orders of magnitude under different n values.

8.5 Performance versus Parameter Δ

Fig. 14 demonstrates the PGNN query performance under different values of Δ and over different data sets. Recall that, we generate the query set Q by first selecting a random point in the data space and then randomly picking up n (precise) query points within a hyperrectangle (centered at the selected point and with the side length $(1{,}000 \times \Delta)$). Intuitively, parameter Δ can approximately indicate the size of $\mathrm{MBR}(Q)$. With large Δ value, the area covered by $\mathrm{MBR}(Q)$ would be large. In the sequel, we vary parameter Δ from 5 percent to 15 percent, where $\beta = 0.8$, the radius range $[r_{min}, r_{max}] = [0, 15]$, the dimensionality $d = 3$, the query size $n = 4$, the data size $|\mathcal{D}| = 30{,}000$, and the number of samples per object used for probability integration is 100.

When parameter Δ becomes large, higher *wall clock time* is required for the PGNN retrieval. In particular, the $PMBM$ method is more sensitive to Δ, since large $\mathrm{MBR}(Q)$ (i.e., large Δ) would result in lower pruning power. In contrast, the effect of Δ on $PSPM$ is smaller, which can be reflected by smoother trends in the figures. Correspondingly, the *speed-up ratios* of both methods decrease with large Δ. However, we find that our approaches can still efficiently answer PGNN queries with speed-up ratios by three orders of magnitude, which confirms the effectiveness of our pruning methods and efficiency of the proposed query procedure.

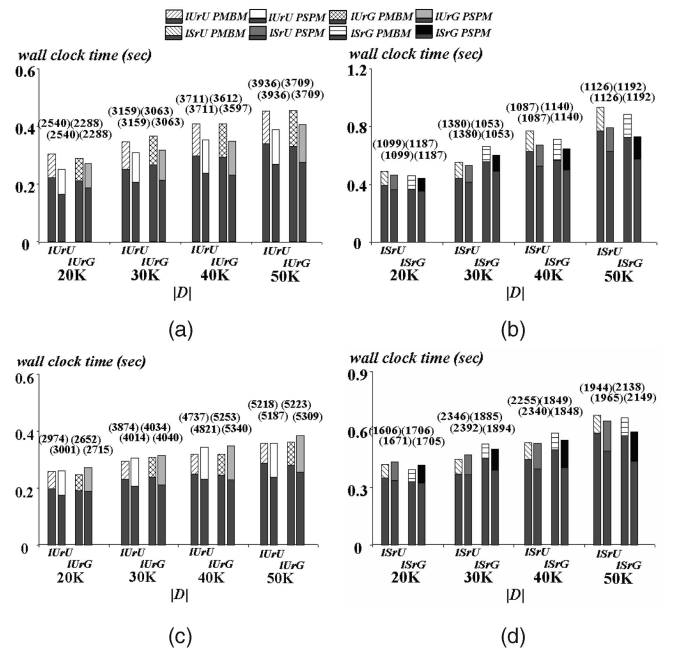


Fig. 15. Scalability test (versus data size $|\mathcal{D}|$, precise query points). (a) $lUrU$ and $lUrG\ (sum)$. (b) $lSrU$ and $lSrG\ (sum)$. (c) $lUrU$ and $lUrG\ (min)$. (d) $lSrU$ and $lSrG\ (min)$.

8.6 Performance versus Data Size $|\mathcal{D}|$

Fig. 15 tests the scalability of our proposed $PMBM$ and $PSPM$ approaches, by varying the data size $|\mathcal{D}|$. Specifically, we evaluate the *wall clock time* of $PSPM$ and $PMBM$ methods over data sets with data size $|\mathcal{D}| = 20{,}000,\ 30{,}000,\ 40{,}000,\ 50{,}000$, where $\beta = 0.8$, the radius range $[r_{min}, r_{max}] = [0, 15]$, the dimensionality $d = 3$, the query size $n = 4$, parameter $\Delta = 10\ \text{percent}$, and the number of samples per object for probability integration is 100.

Table 3 shows the average number of overlapping objects in data sets with different data sizes. When the data size $|\mathcal{D}|$ increases, the number of overlapping objects increases, indicating higher density in the data space. Thus, more objects are identified as PGNN candidates, which results in higher processing cost. That is, the required *wall clock time* for retrieving PGNN candidates increases with the increasing of data size, which can be confirmed in Fig. 15. Furthermore, compared with the *linear scan*, the *speed-up ratio* of our methods $PMBM$ and $PSPM$ remains high (i.e., by three

TABLE 3
Average Number of Overlapping Objects versus Data Size

AVG No. of Overlapping Objects / $\|\mathcal{D}\|$	$lUrU$	$lUrG$	$lSrU$	$lSrG$
$20K$	0.40402	0.340817	4.28631	3.75279
$30K$	0.60742	0.50035	6.32961	5.72392
$40K$	0.81522	0.673617	8.44936	7.74979
$50K$	1.01426	0.848577	10.5071	9.60767

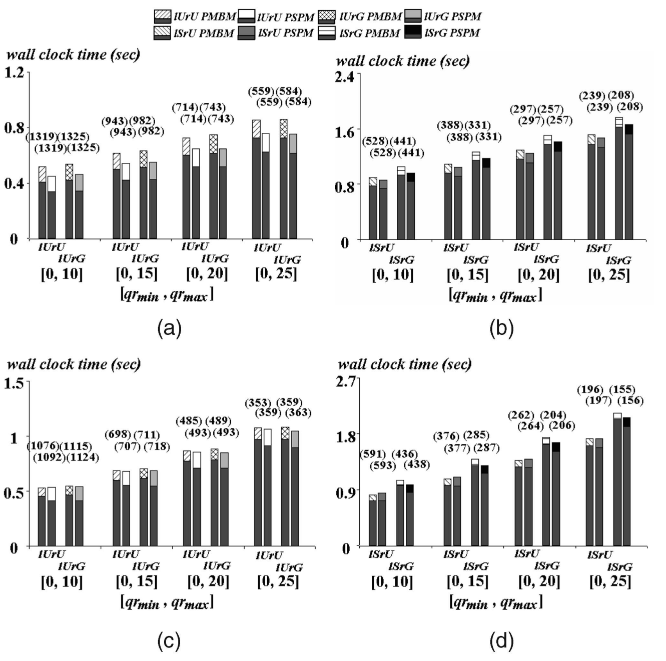


Fig. 16. Performance versus $[qr_{min}, qr_{max}]$ (uncertain query objects). (a) $lUrU$ and $lUrG$ (sum). (b) $lSrU$ and $lSrG$ (sum). (c) $lUrU$ and $lUrG$ (min). (d) $lSrU$ and $lSrG$ (min).

orders of magnitude), which indicates a good scalability of our methods with respect to the data size.

8.7 Query Performance with Uncertain Query Objects

Up to now, we have studied the PGNN query performance under various parameter settings where the query points are precise. We now test the PGNN query with uncertain query objects. Recall that, we generate the center of each uncertain query object in the same way as that with precise query points. Then, we produce the query radius within interval $[qr_{min}, qr_{max}]$ following the same radius distribution as data objects.

Fig. 16 illustrates the PGNN query performance over four data sets, by letting $[qr_{min}, qr_{max}] = [0, 10], [0, 15], [0, 20]$, or $[0, 25]$, where $\beta = 0.8$, the radius range $[r_{min}, r_{max}] = [0, 15]$, the dimensionality $d = 3$, the query size $n = 4$, parameter $\Delta = 10$ percent, data size $|\mathcal{D}| = 30{,}000$, and the number of samples per object used for probability integration is 100. From the figures, we find that the *wall clock time* of both $PMBM$ and $PSPM$ increases when $[qr_{min}, qr_{max}]$ varies from [0, 10] to [0, 25]. Compared with the PGNN query with precise query points, due to the uncertainty of query objects, more data objects are identified as PGNN candidates, and smaller *speed-up ratio* is achieved. However, the *speed-up ratio* is still high (i.e., by two-three orders of magnitude).

In summary, extensive experiments have demonstrated the efficiency and effectiveness of our proposed methods, $PMBM$ and $PSPM$, in answering the PGNN query, over different data sets and under various experimental settings.

9 Conclusions

Query processing over *uncertain* data has become increasingly important due to the inherent uncertainty in many real-world data from different applications. Previous works on *uncertain* data query processing have studied queries in the context of uncertain databases, for example, *NN query*, *range query*, *top-k query*, *skyline query*, and *similarity join*. In this paper, we focus on another important query, PGNN, in the uncertain database, which, to the best of our knowledge, no other work has studied before. Specifically, we propose two novel approaches, namely, $PMBM$ and $PSPM$, which integrate effective pruning methods (i.e., *spatial pruning* and *probabilistic pruning*) to facilitate reducing the PGNN search space. Extensive experiments have demonstrated the efficiency and effectiveness of our proposed methods, under various settings.

Acknowledgments

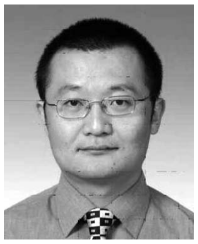
Funding for this work was provided by Hong Kong RGC Grants under Project 611907, National Grand Fundamental Research 973 Program of China under Grant 2006CB303000, NSFC Key Project Grant 60736013, and NSFC Project Grant 60763001.

References

[1] C. Böhm, A. Pryakhin, and M. Schubert, "The Gauss-Tree: Efficient Object Identification in Databases of Probabilistic Feature Vectors," *Proc. 22nd Int'l Conf. Data Eng. (ICDE),* 2006.
[2] L. Chen, M.T. Özsu, and V. Oria, "Robust and Fast Similarity Search for Moving Object Trajectories," *Proc. ACM SIGMOD,* 2005.
[3] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Querying Imprecise Data in Moving Object Environments," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 9, pp. 1112-1127, Sept. 2004.
[4] R. Cheng, D.V. Kalashnikov, and S. Prabhakar, "Evaluating Probabilistic Queries over Imprecise Data," *Proc. ACM SIGMOD,* 2003.
[5] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. Vitter, "Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB),* 2004.
[6] P. Ciaccia, M. Patella, and P. Zezula, "M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces," *Proc. 23rd Int'l Conf. Very Large Data Bases (VLDB),* 1997.
[7] R. Fagin, A. Lotem, and M. Naor, "Optimal Aggregation Algorithms for Middleware," *Proc. 20th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems (PODS),* 2001.
[8] A. Faradjian, J. Gehrke, and P. Bonnet, "Gadt: A Probability Space ADT for Representing and Querying the Physical World," *Proc. 18th Int'l Conf. Data Eng. (ICDE),* 2002.
[9] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," *Proc. ACM SIGMOD,* 1984.
[10] S. Hochreiter, A.S. Younger, and P.R. Conwell, "Learning to Learn Using Gradient Descent," *Proc. Int'l Conf. Artificial Neural Networks (ICANN),* 2001.
[11] E. Hung, Y. Deng, and V.S. Subrahmanian, "RDF Aggregate Queries and Views," *Proc. 21st Int'l Conf. Data Eng. (ICDE),* 2005.
[12] N. Katayama and S. Satoh, "The SR-Tree: An Index Structure for High-Dimensional Nearest Neighbor Queries," *Proc. ACM SIGMOD,* 1997.
[13] H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz, "Probabilistic Similarity Join on Uncertain Data," *Proc. 11th Int'l Conf. Database Systems for Advanced Applications (DASFAA),* 2006.
[14] H.-P. Kriegel, P. Kunath, and M. Renz, "Probabilistic Nearest-Neighbor Query on Uncertain Objects," *Proc. 12th Int'l Conf. Database Systems for Advanced Applications (DASFAA),* 2007.
[15] V. Ljosa and A.K. Singh, "APLA: Indexing Arbitrary Probability Distributions," *Proc. 23rd Int'l Conf. Data Eng. (ICDE),* 2007.
[16] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," *Proc. Fifth Symp. Operating Systems Design and Implementation (OSDI),* 2002.

[17] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis, "Group Nearest Neighbor Queries," *Proc. 20th Int'l Conf. Data Eng. (ICDE),* 2004.
[18] D. Papadias, Y. Tao, K. Mouratidis, and C. Hui, "Aggregate Nearest Neighbor Queries in Spatial Databases," *ACM Trans. Database System,* vol. 30, no. 2, pp. 529-576, 2005.
[19] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic Skylines on Uncertain Data," *Proc. 33rd Int'l Conf. Very Large Data Bases (VLDB),* 2007.
[20] C. Re, N. Dalvi, and D. Suciu, "Efficient Top-k Query Evaluation on Probabilistic Data," *Proc. 23rd Int'l Conf. Data Eng. (ICDE),* 2007.
[21] A.D. Sarma, O. Benjelloun, A.Y. Halevy, and J. Widom, "Working Models for Uncertain Data," *Proc. 22nd Int'l Conf. Data Eng. (ICDE),* 2006.
[22] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-Based Image Retrieval at the End of the Early Years," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 12, pp. 1349-1380, Dec. 2000.
[23] M.A. Soliman, I.F. Ilyas, and K.C. Chang, "Top-k Query Processing in Uncertain Databases," *Proc. 23rd Int'l Conf. Data Eng. (ICDE),* 2007.
[24] Y. Tao, R. Cheng, X. Xiao, W.K. Ngai, and S. Prabhakar, "Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB),* 2005.
[25] Y. Tao, D. Papadias, and X. Lian, "Reverse kNN Search in Arbitrary Dimensionality," *Proc. 30th Int'l Conf. Very Large Data Bases (VLDB),* 2004.
[26] Y. Tao, D. Papadias, X. Lian, and X. Xiao, "Multidimensional Reverse kNN Search," *The VLDB J.,* vol. 16, no. 3, pp. 293-316, 2007.

Xiang Lian received the BS degree from the Department of Computer Science and Technology, Nanjing University, in 2003. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. His research interests include stream time series and probabilistic databases. He is a student member of the IEEE.



Lei Chen received the BS degree in computer science and engineering from Tianjin University, Tianjin, China, in 1994, the MA degree from Asian Institute of Technology, Bangkok, Thailand, in 1997, and the PhD degree in computer science from the University of Waterloo, Waterloo, Ontario, Canada, in 2005. He is currently an assistant professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include multimedia and time series databases, sensor and peer-to-peer databases, and stream and probabilistic databases. He is a member of the IEEE.