# A Generic Framework for Handling Uncertain Data with Local Correlations

Xiang Lian and Lei Chen
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology, Hong Kong, China
{xlian, leichen}@cse.ust.hk

## ABSTRACT

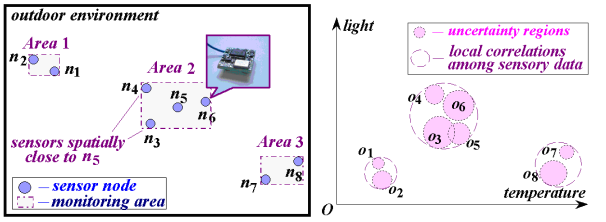
Data uncertainty is ubiquitous in many real-world applications such as sensor/RFID data analysis. In this paper, we investigate uncertain data that exhibit local correlations, that is, each uncertain object is only locally correlated with a small subset of data, while being independent of others. We propose a generic framework for dealing with this kind of uncertain and locally correlated data, in which we investigate a classical spatial query, *nearest neighbor query*, on uncertain data with local correlations (namely LC-PNN). Most importantly, to enable fast LC-PNN query processing, we propose a novel filtering technique via offline pre-computations to reduce the query search space. We demonstrate through extensive experiments the efficiency and effectiveness of our approaches.

## 1. INTRODUCTION

Real-world data often exhibit uncertainty and impreciseness due to the imperfect nature of sensing devices in many applications such as sensor data analysis [10, 6], RFID networks [13], multimedia databases [17], location-based services (LBS) [24], object identification [5], data integration [9], and so on. Figure 1(a) illustrates a sensor network that consists of 8 sensor nodes, $n_1 \sim n_8$, deployed in 3 rectangular monitoring regions ($Areas$ 1, 2, and 3) of an outdoor environment. Figure 1(b) shows the sensory data $o_i$ collected from sensor nodes $n_i$ with 2 attributes, temperature and light. Due to various reasons such as environmental factors, packet loss, and/or low battery power, sensor data (i.e., $o_i$) are often noisy, and the reported data often deviate from the actual values. Such imprecise data can be modeled as uncertain objects residing in *uncertainty regions* (i.e., circles as depicted in Figure 1(b)) in the context of uncertain databases [6]. Within each uncertainty region, the object can reside anywhere following a probabilistic distribution.

In the literature of uncertain databases [6, 25, 22], it is often assumed that uncertain objects are independent. While this independence assumption holds in some real applications, in other scenarios, however, application data may exhibit dependencies and correlate with each other. In the previous example, sensor nodes $n_3 \sim n_6$ in Figure 1(a) are spatially close to each other in a small monitoring region, $Area$ 2. As a result, their collected data $o_3 \sim o_6$


*Proceedings of the VLDB Endowment,* Vol. 4, No. 1


(a) sensor network (b) collected data $o_i$ from sensors $n_i$

**Figure 1: Illustration of Uncertain Data With Local Correlations**

(shown in Figure 1(b)) also tend to be similar and appear correlated with each other. Here, we call these data are *locally correlated*, where the term "locally" means data collected from a sensor are only correlated with a *subset* of the entire sensor data set, as they are independent of other data that are collected from sensors far away, and moreover the "correlated" means that if a sensor reports a value for a parameter (e.g., temperature/light), then its nearby sensors will also report some values with certain probabilities.

In reality, the phenomenon of uncertain objects exhibiting local correlations is not uncommon. For example, noises in the data collected from sensing devices (e.g., RFID readers or sensors) that are spatially close are often correlated. In this case, not all pairs of uncertain objects are dependent (i.e., only locally correlated), that is, RFID/sensor data at sites far away can be independent of each other. Similarly, during the data integration [9], a financial analysis system can collect the trend predictions for stock data from different analysts (i.e., data sources can be their comments or reports about future stocks) via data fusion [4]. Rather than giving fixed predicted values, analysts usually provide estimated uncertain intervals for future stock prices, which can be modeled as uncertain objects. Moreover, different analysts may have the expertise in different areas such as financial or energy sectors. Thus, analysts with the same expertise (e.g., inferred from historical data [9]) tend to give similar (correlated) predictions, while being independent of predicted data from analysts with other expertise. In this case, local correlations exist among the predicted stock data.

Due to the practical existence of data correlation, previous works on query processing that assume independent uncertain objects cannot be accurately used in the correlated case. Furthermore, the *probabilistic graphical model* (PGM) [14] has been adopted in some works [28, 29, 31] to represent probabilistic relational table(s) with globally correlated data (i.e., every pair of uncertain objects are assumed to be correlated). These works usually consider relational queries on these tables, including select, project, and join. In contrast, our work exploits locally correlated uncertain data, and answers advanced spatial queries (including a typical *nearest neighbor query* [27]) on them following a generic framework. While previous works studied basic relational operators on PGM (e.g., with

range predicates on static attributes), their techniques cannot be directly applied to more complex spatial query types, where dynamic distance predicates between objects are involved.

In this paper, we provide a data model for locally correlated data, which generalizes the existing one with the data independence assumption. Typically, to efficiently answer queries on data with local correlations, we investigate a classical spatial query, *probabilistic nearest neighbor query* [6] (namely, LC-PNN), and take it as a case study to illustrate how to enable fast online filtering via offline pre-computations. Our generic framework can also incorporate other spatial queries such as LC-PNN in metric spaces, dynamic skyline [8], and top-$k$ [12] queries, whose guideline can be found in Appendix D. The LC-PNN query has practical applications such as forest surveillance in the GreenOrbs project [21]. GreenOrbs deploys 1,000+ sensor nodes in the forest, collecting various sensory data including temperature, humidity, light, and carbon dioxide titer. Some dangerous events (e.g., fire) in the forest usually correspond to patterns in sensory data. Thus, given a query pattern $q$, the LC-PNN query is useful to find sensors whose (locally correlated and noisy) data are the closest to $q$ with high confidence. Intuitively, locations of the retrieved sensors are the most dangerous places to be taken care of. Similarly, with data integration techniques, these sensory data can be also integrated/fused into other heterogeneous data such as animals' positions (via RFID) in the forest, which are collected, cited, or maintained by different research groups. Due to some of the correlated data sources, the resulting integrated data are locally correlated. The LC-PNN query is thus conducted on such integrated data which can study the habitats and behaviors of animals. Specifically, given a query object $q$ (i.e., a behavior pattern w.r.t. environmental parameters and animals' movement pattern), the LC-PNN query retrieves those animals whose behaviors are highly likely to match with $q$. In the previous stock example, the financial analysis system can issue an LC-PNN query with the actual stock data as query object $q$, and find out those analysts that have the most accurate predictions (e.g., with the smallest Euclidean distances to $q$) with high confidence.

In particular, we highlight our contributions as follows:

1. Our work first models local correlations among uncertain objects in Section 2.1.
2. We present a framework in Section 2.2 for answering queries on locally correlated uncertain data, which generalizes many probabilistic queries with the assumption of object independence, and explore a classical query, LC-PNN in Section 2.3.
3. We propose effective and non-trivial filtering methods in Section 3 to reduce the LC-PNN search space. Especially, a novel cost-model-based offline pre-computation technique is proposed to enable online filtering, which no prior works studied on locally correlated data.
4. We provide an efficient query procedure in Section 4 to answer LC-PNN queries, and provide the guideline for other queries on locally correlated data in Appendix D.

Section 5 evaluates the performance of our approaches. Section 6 overviews previous works on uncertain query processing with and without considering correlations. Section 7 concludes this paper.

# 2. PROBLEM DEFINITION

## 2.1 Data Model

**Uncertain Databases.** An uncertain database $\mathcal{D}^U$ consists of $N$ uncertain objects $o_1$, $o_2$, ..., and $o_N$, where uncertain objects are collected from data sources $n_i$ (e.g., sensor nodes). In the literature of uncertain databases [6], each uncertain object $o_i$ ($1 \leq i \leq N$) can be modeled by an *uncertainty region* $UR(o_i)$, in which $o_i$ resides following arbitrary distribution (see Figure 1(b) as an example). The object distribution can be represented by either *probability density function* (pdf) [6] or discrete samples [25]. In this paper, we adopt the sample representation[1]. Each object $o_i$ has $l$ samples $s_{i,j}$ (or $s_i$ for brevity), where $s_{i,j}$ is associated with an *appearance probability* $s_{i,j}.p \in (0, 1]$ satisfying $\sum_{j=1}^{l} s_{i,j}.p = 1$. Such a representation has practical applications in sensor/RFID networks with the collected data in explicit form of discrete samples[2].

**Local Correlations.** We model the local correlations among uncertain objects $o_i \in \mathcal{D}^U$ as follows. Assume that uncertain database $\mathcal{D}^U$ is composed of several (disjoint) partitions, namely *locally correlated partitions* (LCPs). Each object $o_i \in \mathcal{D}^U$ exactly belongs to one LCP, denoted as $LCS(o_i) \cup \{o_i\}$ (short for *locally correlated set*[3]); objects from the same LCP are locally correlated with each other; and objects from any two distinct LCPs are independent of each other. This model can be used for describing real-world data. For example, Figure 1(b) shows 3 LCPs in the sensor data, where each LCP contains correlated data (uncertain objects) from spatially close sensors deployed in a monitoring area, and data from two areas are independent (since they are far away from each other). In this paper, we consider static uncertain databases, and assume that LCPs can be identified by specific applications.

Note that, our model is generic enough to cover the previously proposed model that assumes either object independence [6, 25] or global correlations [28, 29, 31, 16]. Specifically, they, respectively, correspond to two special cases: 1) the database has multiple LCPs, each containing only one object, and 2) the database has only one LCP, which contains all objects correlated with each other.

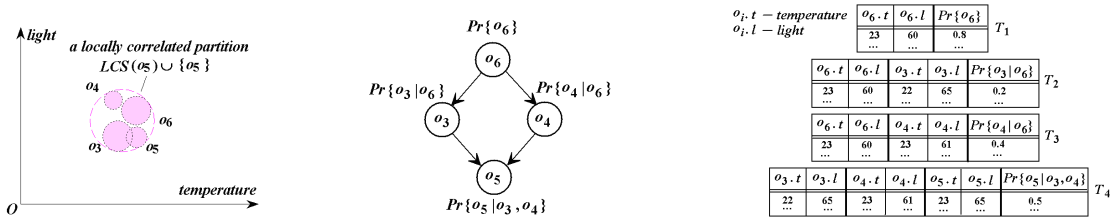
To represent local correlations of objects within an LCP, we adopt one of popular probabilistic graphical models (PGMs) [14], *Bayesian Networks*. Figure 2 illustrates the graphical model for locally correlated uncertain objects in $Area2$ of Figure 1(b). In particular, in Figure 2(a), $o_5$ is locally correlated with other objects in $LCS(o_5) = \{o_3, o_4, o_6\}$. The local correlations of objects in $LCS(o_5) \cup \{o_5\}$ can be expressed by a *directed acyclic graph* (DAG) in Figure 2(b), where each vertex corresponds to an object, and each directed edge, $\overrightarrow{o_i o_j}$, indicates that $o_j$ depends on $o_i$. As an example, edge $\overrightarrow{o_6 o_3}$ in Figure 2(b) implies that, if $o_6$ has certain value pair, e.g., $(temperature, light) = (23, 60)$, then it can be inferred that $o_3$ is also likely to report some values, e.g., $(22, 65)$, with certain probability (e.g., 0.2). To store correlations among vertices (objects), Figure 2(c) summarizes them in 4 *conditional probability tables* (CPTs). That is, Table $T_1$ corresponds to the distribution of attributes $(temperature, light)$ in object $o_6$; similarly, Table $T_2$ ($T_3$) records conditional probabilities in vertex $o_3$ ($o_4$), $Pr\{o_3|o_6\}$ ($Pr\{o_4|o_6\}$); further, Table $T_4$ stores conditional probabilities in vertex $o_5$, that is, $Pr\{o_5|o_3, o_4\}$ (since $o_5$ is correlated with $o_3$ and $o_4$). Note that, in practice, DAG in Figure 2(b) and CPTs in Figure 2(c) are derived by classical theory of the graphical model [14], which (including object uncertainties) are inputs of our query processing framework given later in Section 2.2.

**Queries on Locally Correlated Data.** Therefore, each LCP can be modeled by a local graph (similar to Figure 2), on which queries can be performed over LCPs. While previous works on graphical model [31] usually consider relational operators such as selection, in this paper, we will study more complex spatial queries such as *nearest neighbor queries* (NN).

[1] For the pdf representation, we can first obtain random samples from pdf(s) via sampling techniques, and then apply our proposed methodology in this paper.

[2] If samples are not available, then domain-specific knowledge in real applications (e.g., [5]) can help model the data uncertainty.

[3] Here, we assume that $LCS(o_i)$ excludes $o_i$, though $o_i$ correlates with itself.




(a) Example in Figure 1 (b) DAG Representation (c) Conditional Probability Tables (CPTs)

**Figure 2: An Example of Probabilistic Graphical Model for Uncertain Data with Local Correlations**

For previous works with relational operators, query predicates are usually directly imposed on static attribute values, for example, to compute the joint probability $Pr\{o_3, o_6\}$ that both $o_3$ and $o_5$ report temperature within $[20, 30]$ degrees. Some offline optimizations can be used to return attribute values within the interval and their confidences. That is, such joint probabilities can be obtained by *variable elimination* over CPTs based on the theory of graphical model. In the example of Figure 2(c), both variables $o_3$ and $o_5$ are in Table $T_4$. Thus, it is only needed to eliminate variable $o_4$ in Table $T_4$ by projecting on attributes of $o_3$ and $o_5$ (ignoring $o_4$) and summing up probabilities with the same (projected) attribute values. For joint probabilities with variables in multiple CPTs, variable elimination can be accomplished over the join result of CPTs.

In contrast, our work involves more complicated and computationally intensive spatial queries, which cannot directly borrow techniques for relational operators. Taking the NN query as an example, given a query point $q$, we want to obtain those uncertain objects similar to $q$ with high probability. Different from relational operators, the NN query now involves variables of (correlated) dynamic distances (rather than static attributes) from uncertain objects to an ad-hoc query point $q$. Thus, to find NNs on locally correlated data, the straightforward method is to exhaustively access/join all CPTs, and compute (correlated) distances from samples of all uncertain objects to $q$ (joint probabilities as well). However, this method needs to visit all CPTs, which is more time-consuming than relational operators, in terms of distance and probability computations. As an example in Figure 2, we have to join all the four CPTs, $T_1 \sim T_4$, online obtaining correlated object distances to $q$, and calculate probabilities that objects are NNs. What is worse, due to the ad-hoc $q$, offline optimizations for relational operators on static tables [31] cannot be directly applied to dynamically changing distance variables. Thus, it is more challenging to conduct spatial queries on locally correlated uncertain data.

## 2.2 Generic Framework

We highlight the generic framework for answering spatial queries over uncertain and locally correlated data in three phases:

- **Indexing phase:** We use an R*-tree [1], $\mathcal{I}$, to index locally correlated uncertain data. We bound all objects $o_i \in LCP_j$ (i.e., uncertainty regions $UR(o_i)$) by a *minimum bound rectangle* (MBR), and then insert $LCP_j$ into $\mathcal{I}$ by using the standard "insert" operator of the R-tree. Each LCP is also associated with a graphical representation (e.g., Figure 2).
- **Filtering phase:** For each spatial query $Q$, we traverse the index $\mathcal{I}$, and apply pruning methods to filter out false alarms.
- **Refinement phase:** We refine candidates by checking query predicates via graphical model, and return final answers.

## 2.3 Definition of LC-PNN Queries

In this paper, we particularly study one classical spatial query, nearest neighbor query, on locally correlated uncertain data (called LC-PNN), different from NN on independent uncertain objects [6].

DEFINITION 2.1. (*Probabilistic Nearest Neighbor Query on Locally Correlated Data, LC-PNN*) *Given an uncertain database $\mathcal{D}^U$ with locally correlated uncertain objects, a query point $q$, and a probabilistic threshold $\alpha \in (0, 1]$, an LC-PNN query retrieves those uncertain objects $o_i \in \mathcal{D}^U$ are the nearest neighbors of $q$ with probabilities, $Pr_{LC\text{-}PNN}(q, o_i)$, not smaller than $\alpha$, that is,*

$$Pr_{LC\text{-}PNN}(q, o_i) = \int_{r_1}^{r_2} \left( Pr\{dist(q, o_i) = r\} \cdot Pr\left\{ \bigwedge_{\forall u \in LCS(o_i)} dist(q, u) \geq r \mid r \right\} \cdot Pr\left\{ \bigwedge_{\forall v \in \mathcal{D}^U \setminus (LCS(o_i) \cup \{o_i\})} dist(q, v) \geq r \right\} \right) dr \geq \alpha \quad (1)$$

*where $r_1$ and $r_2$ are the minimum and maximum possible distances from query point $q$ to object $o_i$, respectively, and $dist(x, y) = \sqrt{\sum_{j=1}^{d} (x[j] - y[j])^2}$ for d-dimensional objects $x$ and $y$.*

Intuitively, the LC-PNN probability, $Pr_{LC\text{-}PNN}(q, o_i)$, in Definition 2.1 is the joint probability that all other objects (e.g., $u$ and $v$) have distances to $q$ greater than or equal to $r$ ($= dist(q, o_i)$). In particular, since $o_i$ is correlated with objects $u$ in the same LCP as $o_i$ (i.e., $LCS(o_i)$), given that distance $dist(q, o_i)$ equals to $r$, we can obtain the joint probability that all objects $u \in LCS(o_i)$ have distances to $q$ greater than or equal to $r$ (i.e., the second term in Inequality (1)); similarly, for objects $v$ in other LCPs, we can also obtain their joint probabilities having distances to $q$ not below $r$ (i.e., the third term in Inequality (1)). Due to the independence among distinct LCPs, we multiply the two joint probabilities (terms) discussed above[4]. An object $o_i \in \mathcal{D}^U$ is an LC-PNN, if its LC-PNN probability is not below threshold $\alpha$.

Previous work on NN over independent uncertain data [6] can be subsumed by Definition 2.1. That is, in the case of data independence, the joint probabilities in Inequality (1) can be replaced with multiplications of probabilities that objects have distances to $q$ not smaller than $r$. In our LC-PNN problem, due to the locally correlated data, the processing of LC-PNN is more complex and challenging, which requires computing joint probabilities via graphical model at high costs. A natural nested-loop method would incur high complexity (i.e., $O(N^2)$ for database size $N$). This inspires us to design effective filtering methods, and essentially achieve low LC-PNN retrieval cost by quickly pruning false alarms.

# 3. HEURISTICS OF FILTERING METHODS

## 3.1 Index Pruning

Recall from Section 2.2 that, we store LCPs in an index $\mathcal{I}$ (e.g., R*-tree [1]), which facilitates fast query processing by directly accessing the data we want. Thus, to answer a spatial query like LC-PNN (Definition 2.1) on $\mathcal{I}$, we propose an index pruning method to decide whether or not a node/object can be safely pruned.

LEMMA 3.1. (*Index Pruning, IP*) *Let $best\_so\_far$ be the smallest maximum distance from query point $q$ to any uncertain objects we have seen so far. A node/object $e \in \mathcal{I}$ can be safely pruned, if it*

[4]For the latter probability, if set $\mathcal{D}^U \setminus (LCS(o_i) \cup \{o_i\})$ contains more than one LCP, then it can be also represented by a multiplication of joint probabilities (with distances to $q$ greater than or equal to $r$) from different LCPs.

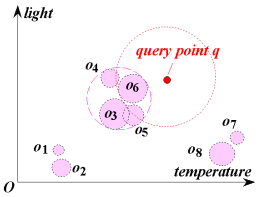



**Figure 3: Heuristics of Candidate Filtering**

*holds that* $mindist(q, e) > best\_so\_far$, *where* $mindist(q, e)$ *is minimum possible distance from query point* $q$ *to node/object* $e$.

**Proof.** Please refer to Appendix A. □

Note that, the pruning condition given in Lemma 3.1 prunes those uncertain objects that definitely cannot be nearest neighbor (NN) of query point $q$. This is similar to the classical NN retrieval over the R-tree index in spatial databases [27] (however, without considering data uncertainty). Similar pruning conditions were given in some existing works [6] for retrieving probabilistic nearest neighbors (PNN) under the assumption of object independence. In contrast, Lemma 3.1 proves the correctness of this pruning condition for LC-PNN queries on locally correlated uncertain data. The index pruning, however, may still produce many LC-PNN candidates (as later shown in experiments of the $Basic$ method in Figure 8), which incurs high refinement cost via the graphical model. This inspires us to propose a novel filtering technique below that utilizes the properties of data local correlation to further reduce the size of the candidate set.

## 3.2 Candidate Filtering via Pre-Computations

We notice that the online computation cost of the LC-PNN probability $Pr_{LC\text{-}PNN}(q, o_i)$ in Inequality (1) is high (i.e., joining CPTs as mentioned in Section 2.1). Thus, we propose a candidate filtering technique which derives and utilizes an upper bound, denoted as $UB\_Pr_{LC\text{-}PNN}(q, o_i)$, of the LC-PNN probability at a lower cost. The rationale is that an object $o_i$ can be safely discarded, if it holds that $UB\_Pr_{LC\text{-}PNN}(q, o_i) < \alpha$. The lemma below guarantees the correctness of this pruning.

LEMMA 3.2. (*Candidate Pruning, CP*) *Let* $UB\_Pr_{LC\text{-}PNN}(q, o_i)$ *be an upper bound of the LC-PNN probability,* $Pr_{LC\text{-}PNN}(q, o_i)$, *given in Inequality (1). Then, uncertain object* $o_i$ *can be safely pruned, if it holds that* $UB\_Pr_{LC\text{-}PNN}(q, o_i) < \alpha$.

**Proof.** Please refer to Appendix B. □

Next, we describe how to obtain this probability upper bound using a 2D example in Figure 3, where $q$ and $o_5$ are query point LC-PNN candidate, respectively. In particular, by overestimating the third term in Inequality (1) as 1, we can derive the LC-PNN upper bound probability for candidate $o_5$:

$$
\begin{aligned}
& Pr_{LC\text{-}PNN}(q, o_5) \qquad (2)\\
&= \int_{r_1}^{r_2} \left( Pr\{dist(q, o_5) = r\} \cdot Pr\left\{ \bigwedge_{\forall u \in LCS(o_5)} dist(q, u) \geq r \mid r \right\} \right.\\
&\quad \left. \cdot\ Pr\left\{ \bigwedge_{\forall v \in \mathcal{D}^U \setminus (LCS(o_5) \cup \{o_5\})} dist(q, v) \geq r \right\} \right) dr\\
&\leq \int_{r_1}^{r_2} \left( Pr\left\{ (dist(q, o_5) = r) \bigwedge \left( \bigwedge_{\forall u \in LCS(o_5)} dist(q, u) \geq r \right) \right\} \cdot 1 \right) dr
\end{aligned}
$$

Below, we use the pre-computed information of LC-PNN candidates to enable the pruning. Specifically, we observe that the Euclidean distance function $dist(\cdot, \cdot)$ follows the *triangle inequality*: $dist(x, y) + dist(y, z) \geq dist(x, z)$.

Our basic idea is to select some pivot, say $piv_{s_i}$, for a sample $s_i$ of LC-PNN candidate $o_i$ ($o_5$ with pivot $piv_{s_5}$ in our example). Then, since it holds that $dist(q, piv_{s_5}) + dist(piv_{s_5}, u) \geq dist(q, u)$ by triangle inequality, we can rewrite Inequality (2) as:

$$
\begin{aligned}
& Pr_{LC\text{-}PNN}(q, o_5) \qquad (3)\\
&\leq \int_{r_1}^{r_2} \left( Pr\left\{ (dist(q, o_5) = r) \bigwedge \left( \bigwedge_{\forall u \in LCS(o_5)} dist(q, u) \geq r \right) \right\} \right) dr\\
&\leq \int_{r_1}^{r_2} \left( Pr\left\{ (dist(q, piv_{s_5}) + dist(piv_{s_5}, o_5) \geq r) \bigwedge \right.\right.\\
&\qquad \left.\left. \left( \bigwedge_{\forall u \in LCS(o_5)} dist(q, piv_{s_5}) + dist(piv_{s_5}, u) \geq r \right) \right\} \right) dr.\\
&\qquad\qquad \text{// upper bound, } UB\_Pr_{LC\text{-}PNN}(q, o_5)
\end{aligned}
$$

Inequality (3) exactly gives an upper bound, $UB\_Pr_{LC\text{-}PNN}(q, o_5)$, of the LC-PNN probability for candidate $o_5$. One interesting property of this upper bound is that, we can offline pre-compute (an upper bound of) the joint probability, denoted as $JP_o(s_5)$:

$$JP_o(s_5) = Pr\left\{ \bigwedge_{\forall u \in LCS(o_5) \cup \{o_5\}} dist(q, piv_{s_5}) + dist(piv_{s_5}, u) \geq r \right\},$$

in Inequality (3), for $r = dist(q, s_5)$, which is equivalent to:

$$JP_o(s_5) = Pr\left\{ \bigwedge_{\forall u \in LCS(o_5) \cup \{o_5\}} dist(piv_{s_5}, u) \geq dist(q, s_5) - dist(q, piv_{s_5}) \right\}.$$

Observing that, in the joint probability above, the LHS of the inequality, $dist(piv_{s_5}, u)$, can be obtained offline without knowing query point $q$ (once pivot $piv_{s_5}$ is selected), whereas the RHS (i.e., $dist(q, s_5) - dist(q, piv_{s_5})$) is related to query point $q$. Therefore, we only need to offline pre-compute the joint probability above by replacing RHS with some pre-selected constants $\lambda$.

In a general case, for sample $s_i$ of an uncertain object $o_i$, we pre-compute the joint probability, $JP_o(s_i, \lambda)$, defined below.

DEFINITION 3.1. (*Pre-Computed Joint Probabilities,* $JP_o(s_i, \lambda)$) *For any sample* $s_i$ *of an uncertain object* $o_i$, *we define the joint probability,* $JP_o(s_i, \lambda)$, *below.*

$$JP_o(s_i, \lambda) = Pr\left\{ \bigwedge_{\forall u \in LCS(o_i) \cup \{o_i\}} dist(piv_{s_i}, u) \geq \lambda \right\}. \qquad (4)$$

The joint probabilities $JP_o(\cdot)$ above can be pre-computed offline via standard variable elimination in graphical model [14]. For any online LC-PNN query w.r.t. candidate $o_i$ (with sample $s_i$), we first compute distances $dist(q, s_i)$ and $dist(q, piv_{s_i})$. Then, we choose a largest $\lambda$ value (we pre-selected) satisfying $\lambda \leq (dist(q, s_i) - dist(q, piv_{s_i}))$, and use its pre-computed joint probability $JP_o(s_i, \lambda)$ (in Eq. (4)) to derive the LC-PNN probability upper bound below. Here, we use the largest $\lambda$, since it can result in tighter (i.e., smaller) upper bound and achieve higher pruning power.

LEMMA 3.3. (*Upper Bound of LC-PNN Probability*) *Given a query point* $q$ *and an LC-PNN candidate* $o_i$, *the upper bound of the LC-PNN probability,* $UB\_Pr_{LC\text{-}PNN}(q, o_i)$, *is given by:*

$$UB\_Pr_{LC\text{-}PNN}(q, o_i) = \sum_{\forall s_i \in o_i \wedge \lambda = max\{\lambda' \mid \lambda' \leq dist(q, s_i) - dist(q, piv_{s_i})\}} JP_o(s_i, \lambda) \qquad (5)$$

*where* $JP_o(\cdot)$ *is given in Definition 3.1.*

By using pre-computations w.r.t. pivots, to derive the probability upper bound in Lemma 3.3, we only need to online compute distances from $q$ to $s_i$ and $piv_{s_i}$, rather than exhaustively computing distances from $q$ to all objects in LCPs, thus greatly saving the cost.

**Discussions on Obtaining the Domain of $\lambda$.** We address the remaining issue on how to obtain the domain, $[min\_\lambda, max\_\lambda]$, of $\lambda$ values above. Assuming we know the domain of the data space, we can obtain the minimum and maximum possible values of $\lambda$ ($= dist(q, s_i) - dist(q, piv_{s_i})$) for any position of $q$ in the data space. This method, however, may lead to a loose bound of $\lambda$, and

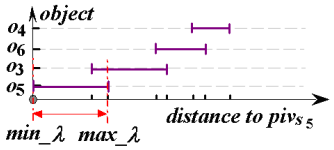



**Figure 4: Illustration of Obtaining $min\_\lambda$ and $max\_\lambda$**

it is not useful for obtaining a tight probability upper bound[5]. Thus, we adopt another direction to obtain domain $[min\_\lambda, max\_\lambda]$.

Instead of considering $\lambda$ on RHS of inequality in Eq. (4), we check the bound of its LHS (i.e., $dist(piv_{s_i}, u)$). Figure 4 shows distance bounds from a pivot $piv_{s_5}$ to objects $u \in \{o_3, o_4, o_5, o_6\}$ in LCP of $o_5$, in the previous example of Figure 3. We can obtain domain of $\lambda$ by letting $min\_\lambda = min_{\forall u \in LCS(o_5)}\{mindist(piv_{s_5}, u)\}$ and $max\_\lambda = min_{\forall u \in LCS(o_5)}\{maxdist(piv_{s_5}, u)\}$.

The reason for the settings above is that: When $\lambda < min\_\lambda$, the joint probability, $JP_o(s_i, \lambda)$, in Eq. (4) is always 1; similarly, when $\lambda > max\_\lambda$, we have $JP_o(s_i, \lambda) = 0$. Thus, when we offline select $\lambda$ values, we only need to choose values within $[min\_\lambda, max\_\lambda]$. The number of selected $\lambda$ values for pre-computation depends on the available space (note: upper bounded by the total number of samples in $LCS(o_i) \cup \{o_i\}$). The pivot selection will be discussed later in Section 4.3, based on a formal cost model.

# 4. QUERY PROCESSING

## 4.1 Candidate Filtering via Nodes

Since LCPs are indexed in an R*-tree (Section 2.2), we will consider pruning candidate $o_i$ by using pre-computed probabilities from an index node $e$ (containing LCPs) which are defined below.

DEFINITION 4.1. (*Pre-Computed Joint Probabilities for Index Nodes*) *Given an index node e, a pivot $piv_e$, and a threshold $\lambda$, we define the joint probability, $JP_e(e, \lambda)$, as follows.*

$$JP_e(e, \lambda) = \prod_{\forall LCP_j \in e} \left( Pr\left\{ \bigwedge_{\forall u \in LCP_j} dist(piv_e, u) \geq \lambda \right\} \right). \quad (6)$$

In Definition 4.1, $JP_e(e, \lambda)$ is offline pre-computed, which is the joint probability that all the objects $u$ in node $e$ have distances to a node pivot $piv_e$ greater than or equal to $\lambda$. Similar to candidate filtering in Section 3.2, we can derive an upper bound of the LC-PNN probability via $JP_e(e, \lambda)$ for a candidate $o_i$.

LEMMA 4.1. (*Upper Bound of LC-PNN Probability via Nodes*) *Given a query point q, an LC-PNN candidate $o_i$, an index node e, and a pivot $piv_e$, the upper bound of the LC-PNN probability, $UB\_Pr_{LC\text{-}PNN}(q, o_i)$, is given by:*

$$UB\_Pr_{LC\text{-}PNN}(q, o_i) = \sum_{\forall s_i \in o_i \wedge \lambda = max\{\lambda' | \lambda' \leq dist(q, s_i) - dist(q, piv_e)\}} JP_e(e, \lambda) \quad (7)$$

*where $JP_e(\cdot)$ is given in Definition 4.1.*

Once we obtain the probability upper bound $UB\_Pr_{LC\text{-}PNN}(q, o_i)$, we can check the pruning condition in Lemma 3.2. Note that, for any node $e$, we simply select the corner points of $e$ as pivots $piv_e$ in Eq. (6). Moreover, similar to discussions on parameter $\lambda$ in Section 3.2, the domain of $\lambda$ in $JP_e(\cdot)$ of Eq. (6) can be given by $[min\_\lambda, max\_\lambda]$, where $min\_\lambda = min_{\forall u \in e}\{mindist(piv_e, u)\}$ and $max\_\lambda = min_{\forall u \in e}\{maxdist(piv_e, u)\}$.

## 4.2 LC-PNN Query Procedure

The LC-PNN query processing can be achieved by traversing the index that contains locally correlated uncertain data. Specifically, we traverse the index in a best-first manner [11], and apply either index pruning or candidate filtering via pre-computations discussed in Sections 3 and 4.1. If a node/LCP cannot be pruned, then we need to further check its children; if an object cannot be pruned, then this object is an LC-PNN candidate. Finally, we refine candidates and return true LC-PNN answers. Please refer to the detailed pseudo-code of LC-PNN query processing in Appendix C.

[5]The bound can be too loose w.r.t. all possible positions of $q$, while $q$ expects to be close to LC-PNN candidate $o_5$ (i.e., $dist(q, s_5) - dist(q, piv_{s_5})$ should be small).

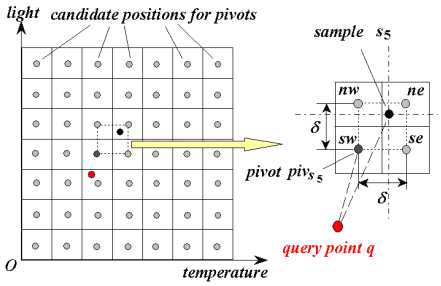



**Figure 5: Illustration of Pivot Selection**

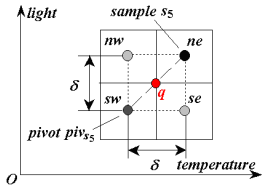



**Figure 6: Illustration of Deciding $\delta_{max}$**

## 4.3 Cost Model for Selecting Pivots

Next, we illustrate how to select pivots to enable effective candidate filtering. Recall from Section 3.2 that, we use the LC-PNN probability upper bound (given in Eq. (5)) to prune candidate $o_i$ (in case $UB\_Pr_{LC\text{-}PNN}(q, o_i) < \alpha$). Our goal is to select "good" pivots such that this upper bound is as small (i.e., close to the actual LC-PNN probability) as possible, achieving high pruning power.

We use an example in Figure 5 to illustrate our basic idea of selecting pivots. Specifically, we partition the data space into cells of equal size (i.e., $\delta \times \delta$), and the center of each cell is considered as a candidate position for pivot[6]. Consider a sample $s_5$ of object $o_5$ in a rectangle with four candidate positions ($ne$, $nw$, $sw$, and $se$) of pivots as corners. Given a query point $q$, our intuition is to select a pivot $piv_{s_5}$ such that the upper bound probability given by Eq. (4) at least has some pruning power. In other words, it is desirable that $dist(q, s_5) - dist(q, piv_{s_5}) > 0$ (i.e., $\lambda > 0$ in Eq. (4)). Based on this heuristics, we always select a position for pivot (from four candidate positions) that is in the same quadrant (w.r.t. sample $s_5$) as $q$. Thus, in Figure 5, for sample $s_5$, we can choose $sw$ as pivot $piv_{s_5}$, and offline pre-compute probability upper bounds in Eq. (4) using 4 candidate positions. For any query point $q$, we can decide its position in the quadrant of $s_5$, and use the pre-computed bound corresponding to the pivot in that quadrant. This way, the filtering over LC-PNN candidate can be performed.

Now the only remaining issue is on how to set the side length, $\delta$, of cells. In the sequel, we first give the maximum possible $\delta$ value, denoted as $\delta_{max}$. Then, we provide a cost model for deciding a $\delta$ value within $[0, \delta_{max}]$ that achieves low query processing cost.

### 4.3.1 Deciding $\delta_{max}$

As mentioned earlier, to guarantee the pruning power of probability upper bound, it is desired that $dist(q, s_5) - dist(q, piv_{s_5}) > 0$ holds. Figure 6 exactly corresponds to the extreme extreme case with critical condition $dist(q, s_5) - dist(q, piv_{s_5}) = 0$, where sample $s_5$ resides at point $ne$, and query point $q$ be located at the mid-

[6]For highly skewed data, we can divide data space into smaller partitions, and set $\delta$ or pivots for each partition.

| Parameters | Values |
|---|---|
| $\alpha$ | $0.1, 0.2, \mathbf{0.5}, 0.8, 0.9$ |
| $[e_{min}^{LCP}, e_{max}^{LCP}]$ | $[1,1], [1,2], [\mathbf{1},\mathbf{3}], [1,4], [1,5]$ |
| $N$ | $50K, 100K, \mathbf{150K}, 200K, 250K$ |
| $n$ | $2, 3, \mathbf{5}, 8, 10$ |

**Table 1: The Parameter Settings**

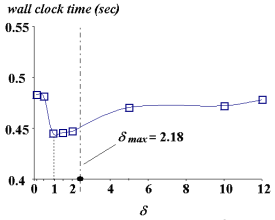



**Figure 7: Selection of Parameter $\delta$ for Pivots ($lUeU$)**

dle point between $ne$ and $sw$. Clearly, if the diagonal length (i.e., $\sqrt{2}\delta$ in the 2D case) of the rectangle (with $sw$ and $ne$ as corners) is greater than $2{\cdot}dist(q, s_5)$, then we have $dist(q, s_5){-}dist(q, piv_{s_5})$ $<0$, and thus the LC-PNN probability is 1, having no pruning power. So we should let $\sqrt{2}\cdot\delta \leq 2\cdot dist(q, s_5)$ in the 2D space. In a general $d$-dimensional space, for sample $s_i$ of object $o_i$, we have:

$$\delta \leq \frac{2\cdot dist(q, s_i)}{\sqrt{d}} \leq \frac{2\cdot (dist(q, C_{o_i}) + r_{o_i})}{\sqrt{d}}. \quad (8)$$

where $C_{o_i}$ and $r_{o_i}$ are the center and radius of uncertainty region $UR(o_i)$ for object $o_i$, respectively.

Therefore, we can obtain $\delta_{max}$ by estimating the RHS of Inequality (8). For $r_{o_i}$, we can approximate it by the average radius of uncertain objects in the database. Moreover, observing that object $o_i$ is an LC-PNN candidate close to query point $q$, we thus estimate $dist(q, C_{o_i})$ as the distance from $q$ to its nearest center among all centers of uncertain objects. In other words, in a hypersphere centered at $q$ with radius $dist(q, C_{o_i})$, there exists only one center point $C_{o_i}$. Thus, we apply the *power law* [2], resulting in:

$$(N-1)\cdot\left(\frac{2\pi^{d/2}\cdot dist^d(q, C_{o_i})}{d\cdot\Gamma(d/2)}\right)^{D_2/d} = 1, \quad (9)$$

which can be simplified as:

$$dist(q, C_{o_i}) = \left(\frac{(d\cdot\Gamma(d/2))^{D_2/d}}{(2\pi^{d/2})^{D_2/d}\cdot(N-1)}\right)^{1/D_2}, \quad (10)$$

where $D_2$ is the *correlation fractal dimension* of all the object centers in $\mathcal{D}^U$, $d$ is the dimensionality of the database, and $\Gamma(\cdot)$ is the *gamma function*. By substituting $dist(q, C_{o_i})$ in Eq. (10) and average radius $r_{o_i}$ into RHS of Eq. (8), we obtain the $\delta_{max}$ value. Note that, since power law can capture arbitrary distribution, our cost model expects to achieve low query cost.

### 4.3.2 Selection of $\delta$

We next decide which value in $[0, \delta_{max}]$ is appropriate for setting parameter $\delta$. Below, we provide a cost model to formalize the query cost, involving parameter $\delta$, and aim to choose a good $\delta$ value such that this query cost is minimized. Specifically, the query cost consists of two parts, the cost of computing distances between $q$ and pivots, $cost_{filter}$, and that of refining candidates, $cost_{refine}$. Cost $cost_{filter}$ is proportional to the number of possible candidate pivots within the region where LC-PNN candidates reside. By applying the *power law* [2], we can obtain:

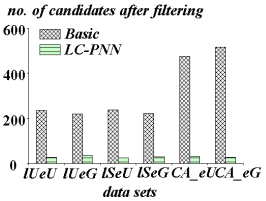
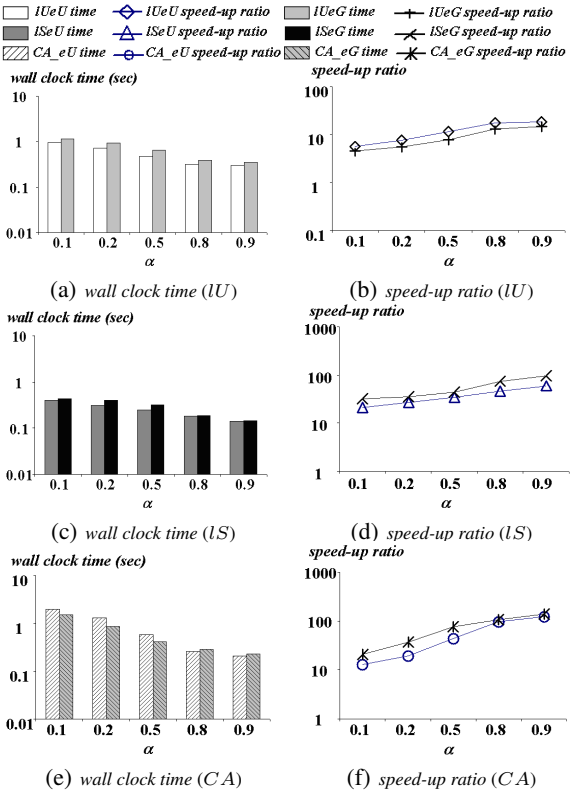



**Figure 8: Comparison of Filtering Power ($Basic$ vs. $LC$-$PNN$)**




(a) *wall clock time* ($lU$) (b) *speed-up ratio* ($lU$)

(c) *wall clock time* ($lS$) (d) *speed-up ratio* ($lS$)

(e) *wall clock time* ($CA$) (f) *speed-up ratio* ($CA$)

**Figure 9: Performance vs. Probabilistic Threshold $\alpha$**

$$cost_{filter} = (N-1)\cdot\left(\frac{2\pi^{d/2}\cdot(dist(q, C_{o_i}) + 2\cdot r_{o_i})^d}{d\cdot\Gamma(d/2)}\right)^{D_2/d} \cdot\left(\frac{2\pi^{d/2}\cdot(r_{o_i}+\delta)^d/\delta^d}{d\cdot\Gamma(d/2)}\right)\cdot t_{flt} \quad (11)$$

In Eq. (11), the first two terms estimate the number of LC-PNN candidates by the power law, the third term is the number of cells (pivots) that are relevant to each candidate, and the fourth one $t_{flt}$ is the unit time cost to check the filtering condition via pivots.

The second refinement cost, $cost_{refine}$, is related to the pruning power of pivot. Let $PP$ be the probability that an object $o_i$ can be pruned by our filtering method, that is,

$$PP = Pr\left\{Pr\left\{\bigwedge_{\forall u\in LCS(o_5)\cup\{o_5\}} dist(piv_{s_5}, u) \geq \lambda\right\} < \alpha\right\}. \quad (12)$$

Therefore, we have the refinement cost as follows:

$$cost_{refine} = (N-1)\cdot\left(\frac{2\pi^{d/2}\cdot(dist(q, C_{o_i}) + 2\cdot r_{o_i})^d}{d\cdot\Gamma(d/2)}\right)^{D_2/d} \cdot(1-PP)\cdot t_{rfn}, \quad (13)$$

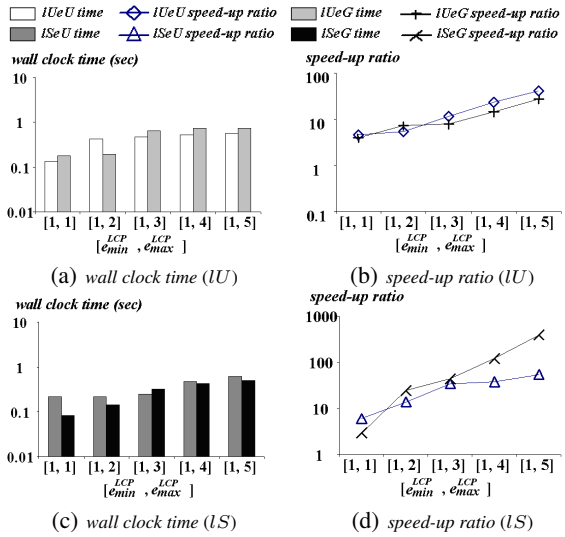
where the first three terms estimate the number of candidates after filtering, and the fourth one $t_{rfn}$ is the average time cost (obtained from statistics of query history) to refine a candidate.

This way, we can estimate the query cost, $cost_{filter}{+}cost_{refine}$, for any $\delta$ in $[0, \delta_{max}]$, and select the one with the lowest cost as $\delta$.

**Discussions on Other Queries Over Locally Correlated Data.** Please refer to Appendix E for other query types in our framework.

## 5. EXPERIMENTAL EVALUATION

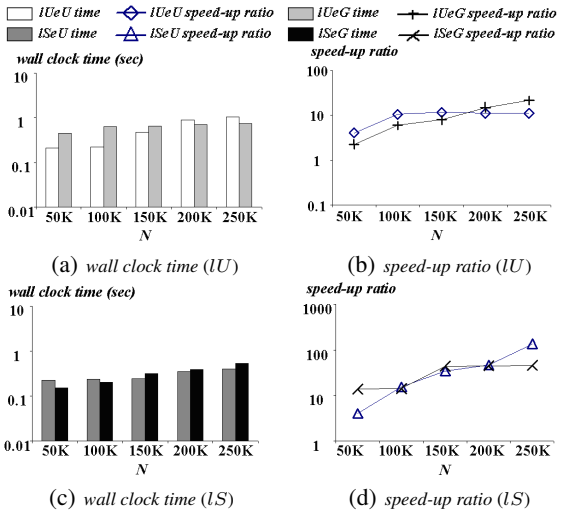
**Experimental Settings.** We conduct experiments on real/synthetic locally correlated uncertain data. For synthetic data, each LCP, $LCP_j$, is centered at $C_{LCP_j}$ with extent $e_{LCP_j}{\in}[e_{min}^{LCP}, e_{max}^{LCP}]$ on each dimension, in which we produce locally correlated data. For different center and extent distributions, we obtain 4 data sets $lUeU$, $lUeG$, $lSeU$, and $lSeG$. We also test real California Road Network data, $CA\_eU$ and $CA\_eG$. The detailed data descriptions




(a) *wall clock time* ($lU$) (b) *speed-up ratio* ($lU$)

(c) *wall clock time* ($lS$) (d) *speed-up ratio* ($lS$)

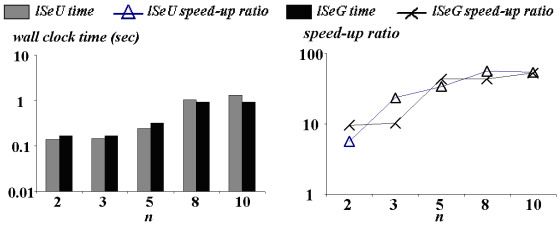
**Figure 10: Performance vs. LCP Extent Range** $[e_{min}^{LCP}, e_{max}^{LCP}]$




(a) *wall clock time* ($lU$) (b) *speed-up ratio* ($lU$)

(c) *wall clock time* ($lS$) (d) *speed-up ratio* ($lS$)

**Figure 11: Performance vs. Data Size** $N$




(a) *wall clock time* ($lS$) (b) *speed-up ratio* ($lS$)

**Figure 12: Performance vs. LCP Size** $n$

are in Appendix E. We compare our approach, $LC\text{-}PNN$, with a modified index-based method [6], $Basic$, which obtains PNN candidates via index, and refines candidates by considering data correlations rather than independence. We report the *wall clock time*, which is the execution time of LC-PNN, and *speed-up ratio*, defined as the time cost of $Basic$ divided by that of $LC\text{-}PNN$. Table 1 depicts parameter settings, where numbers in bold font are *default values*. Each time we vary one parameter, while setting other parameters to default values. All our experiments are conducted on a Pentium IV 3.2GHz PC with 1G memory.

**Selection of Parameter $\delta$.** Figure 7 shows the effect of parameter $\delta$ on the $LC\text{-}PNN$ performance over data set $lUeU$, where $\delta$ is a parameter mentioned in Section 4.3 to determine positions of pivots. Large $\delta$ results in low computation cost via pivots, however, low pruning power as well. Thus, there is a trade-off between computation cost and pruning power. In Figure 7, when $\delta$ increases, the total time cost first decreases and then increases. The maximum value, $\delta_{max}$, of $\delta$ is given by 2.18, derived by our cost model (Section 4.3.1). When $\delta = 1$, the time cost achieves the lowest value. The trends with other data sets are similar and thus omitted. In subsequent experiments, in order to show the trend of other parameters without being affected by different $\delta$, we will set $\delta$ to 1.

**Comparison of Filtering Power.** Figure 8 compares $LC\text{-}PNN$ with $Basic$ on 6 real/synthetic data, in terms of the number of candidates after filtering, where parameters are set to their default values. We can see that $LC\text{-}PNN$ incurs much fewer (by orders of magnitude) candidates than $Basic$, which confirms the effectiveness of our candidate filtering technique via offline pre-computations.

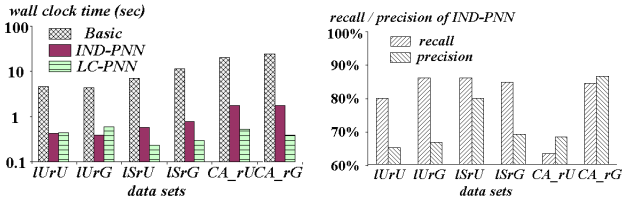
**Effect of Probabilistic Threshold $\alpha$.** Figure 9 illustrates the effect of $\alpha$ on the LC-PNN performance, with $\alpha$ varying from 0.1 to 0.9. Larger $\alpha$ results in higher pruning power due to the candidate filtering. In figures, the wall clock time of $LC\text{-}PNN$ is low (i.e., around 0.45$\sim$1 sec for default settings), and decreases for large $\alpha$, due to the retrieval and refinement of fewer candidates. In contrast, $Basic$ obtains candidates without considering $\alpha$ constraint. Thus, the speed-up ratios increase for large $\alpha$. The results on real data $CA$ are similar, thus, we only show results on synthetic data below.

**Effect of LCP Extent Range $[e_{min}^{LCP}, e_{max}^{LCP}]$.** Figure 10 presents results by varying $[e_{min}^{LCP}, e_{max}^{LCP}]$ from $[1, 1]$ to $[1, 5]$. When the range is wider, the wall clock time slightly increases. This is because more LCPs are accessed and filtered for wider range, which requires higher cost. Nonetheless, the time cost is low (i.e., 0.1$\sim$1 sec). The speed-up ratio of $LC\text{-}PNN$ increases with wider range, as $Basic$ produces more candidates than $LC\text{-}PNN$.

**Effect of Data Size $N$.** Figure 11 tests the scalability of $LC\text{-}PNN$ on the total number, $N$, of uncertain objects from $50K$ to $250K$. In figures, the wall clock time of $LC\text{-}PNN$ smoothly increases with the increasing $N$. Nonetheless, the wall clock time remains low (i.e., about 1 sec). Further, the speed-up ratio indicates that $LC\text{-}PNN$ outperforms $Basic$ by 1-2 orders of magnitude, which shows good scalability of our $LC\text{-}PNN$ approach.

**Effect of LCP Size $n$.** Figure 12 evaluates the wall clock time and speed-up ratio of $LC\text{-}PNN$ on $lS$ data sets by varying the average number of objects per LCP, $n$, from 2 to 10. The results on $lU$ data are similar and omitted. Due to larger LCP size, the calculation of joint probability via the graphical model becomes more costly; moreover, since the total data size increases, the data retrieval leads to more I/O and computation costs. Thus, in Figure 12(a), the wall clock time increases for larger $n$. Further, for larger $n$, due to our proposed filtering technique via pre-computations, $LC\text{-}PNN$ has much fewer candidates than $Basic$. Thus, the speed-up ratio increases with the increasing $n$, and remains high.

**Comparisons with PNN With Object Independence.** We compare $Basic$ and $LC\text{-}PNN$ with PNN over independent uncertain objects [6], namely $IND\text{-}PNN$, which has the same index traversal as $Basic$, but in the refinement step replaces the joint probability in Inequality (1) with multiplications of probabilities for individual objects ($v$ or $u$). Figure 13(a) shows the efficiency comparison on real/synthetic data. $IND\text{-}PNN$ has much lower time cost than $Basic$, since $IND\text{-}PNN$ assumes independent data and does not need to compute joint probabilities. For $lU$ data, $LC\text{-}PNN$ has a bit higher (but comparable) wall clock time than $IND\text{-}PNN$, due to more refinement cost via graphical model; for $lS$ and $CA$ data, $LC\text{-}PNN$ performs better than $IND\text{-}PNN$, since $LC\text{-}PNN$ has fewer candidates to refine. Figure 13(b) shows the recall and precision of $IND\text{-}PNN$ on locally correlated data, where $LC\text{-}PNN$ answers are the ground truth, recall is the number of true answers obtained by $IND\text{-}PNN$ divided by that of true answers, and precision is the percentage of true answers in the




(a) *wall clock time* (b) *recall / precision of* $IND\text{-}PNN$

**Figure 13: Comparison of** $Basic$**,** $IND\text{-}PNN$**, and** $LC\text{-}PNN$

returned $IND\text{-}PNN$ answer set. About 15% - 40% of $LC\text{-}PNN$ answers are missing by using $IND\text{-}PNN$; moreover, 15% - 35% of $IND\text{-}PNN$ results are not $LC\text{-}PNN$ answers, incurring extra refinement cost, which in turn shows the effectiveness of $LC\text{-}PNN$.

# 6. RELATED WORK

Previous works on uncertain query processing can be classified into two categories, with and without object independence. The first category includes NN query [6, 3], skyline [25], similarity join [18, 23], and so on. In probabilistic databases [7], top-$k$ queries [26, 30, 32, 20] are also explored assuming independence among $x$-*tuples* under *possible worlds* semantics. The most relevant work to ours is the *probabilistic nearest neighbor* (PNN) query [6, 3] on independent uncertain data. Their essential idea is to use boundaries of uncertainty regions to reduce the search space. Then, candidates are refined by assuming object independence. Thus, the correctness and applicability of these techniques to locally correlated uncertain data are not trivial. In our work, we proved the correctness of index pruning and designed an effective candidate filtering method.

The second category considers correlations among uncertain data [28, 29, 31]. Sen and Deshpande [28] presented a framework for explicitly modeling/querying correlated tuples via graphical models in probabilistic databases. Sen et al. [29] exploited shared correlations in uncertain data to speed up the efficiency of relational operators (e.g., join). Wang et al. [31] based on a class of *First-Order Bayesian Network* to model correlated uncertain data, and performed relational operators (e.g., select, project and join). Kanagal and Deshpande [16] indexed correlated probabilistic databases to answer inference/aggregation queries. A large junction tree is recursively divided into partitions via classical tree partition algorithms which form a hierarchical tree structure. While the index in [16] is obtained by partitioning the tree (that may incur high cost of visiting all tree nodes for spatial queries in the Euclidean space like LC-PNN), our work uses the index that naturally stores locally correlated data in Euclidean spaces. Further, the index of [16] stores joint probabilities for variables under/among nodes for selection on static attributes. In our problem, however, spatial queries like LC-PNN have ad-hoc query point only known at query time. To enable lightweight query processing, we have to design pruning methods via offline pre-computations. Other works [15, 19] model temporally correlated probabilistic streams by a simplified graphical model, Markovian Model, whereas our work focuses on spatially locally correlated data; further, our targeting queries are spatial queries, not relational operators. Thus, previous methods are not directly applicable with different data and queries.

# 7. CONCLUSION

Uncertain and locally correlated data are pervasive in many real applications. In this paper, we propose a generic framework for handling such uncertain and locally correlated data. We study a typical query, *probabilistic nearest neighbor query* (LC-PNN), on locally correlated uncertain data. To facilitate fast query processing, we propose a novel filtering technique via offline pre-computations to reduce the search space. We demonstrate through extensive experiments the LC-PNN performance of our approach.

# 8. REFERENCES


[1] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *SIGMOD*, 1990.

[2] A. Belussi and C. Faloutsos. Self-spacial join selectivity estimation using fractal concepts. *ACM Trans. Inf. Syst.*, 16(2), 1998.

[3] G. Beskales, M. Soliman, and I. F. Ilyas. Efficient search for the top-k probable nearest neighbors in uncertain databases. In *VLDB*, 2008.

[4] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1), 2008.

[5] C. Böhm, A. Pryakhin, and M. Schubert. The Gauss-tree: efficient object identification in databases of probabilistic feature vectors. In *ICDE*, 2006.

[6] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, 2003.

[7] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4), 2007.

[8] E. Dellis and B. Seeger. Efficient computation of reverse skyline queries. In *VLDB*, 2007.

[9] X. L. Dong, L. Berti-Equille, and D. Srivastava. Integrating conflicting data: The role of source dependence. *PVLDB*, 2(1), 2009.

[10] A. Faradjian, J. Gehrke, and P. Bonnet. Gadt: A probability space ADT for representing and querying the physical world. In *ICDE*, 2002.

[11] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2), 1999.

[12] V. Hristidis, N. Koudas, and Y. Papakonstantinou. PREFER: A system for the efficient execution of multi-parametric ranked queries. In *SIGMOD*, 2001.

[13] S. R. Jeffery, M. J. Franklin, and M. Garofalakis. An adaptive RFID middleware for supporting metaphysical data independence. *VLDBJ*, 17(2), 2008.

[14] M. I. Jordan. Graphical models. In *Statistical Science (Special Issue on Bayesian Statistics)*, 2004.

[15] B. Kanagal and A. Deshpande. Efficient query evaluation over temporally correlated probabilistic streams. In *ICDE*, 2009.

[16] B. Kanagal and A. Deshpande. Indexing correlated probabilistic databases. In *SIGMOD*, 2009.

[17] M. Koskela, J. Laaksonen, and E. Oja. Use of image subset features in image retrieval with self-organizing maps. In *CIVR*, 2004.

[18] H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz. Probabilistic similarity join on uncertain data. In *DASFAA*, 2006.

[19] J. Letchner, C. Ré, M. Balazinska, and M. Philipose. Access methods for markovian streams. In *ICDE*, 2009.

[20] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1), 2009.

[21] M. Li, Y. He, Y. Liu, J. Zhao, S. Tang, X.-Y. Li, and G. Dai. Canopy closure estimates with greenorbs: Sustainable sensing in the forest. In *ACM Sensys*, 2009. http://greenorbs.org.

[22] X. Lian and L. Chen. Monochromatic and bichromatic reverse skyline search over uncertain databases. In *SIGMOD*, 2008.

[23] V. Ljosa and A. K. Singh. Top-$k$ spatial joins of probabilistic objects. In *ICDE*, 2008.

[24] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *VLDB*, 2006.

[25] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *VLDB*, 2007.

[26] C. Re, N. Dalvi, and D. Suciu. Efficient top-$k$ query evaluation on probabilistic data. In *ICDE*, 2007.

[27] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *SIGMOD*, 1995.

[28] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, 2007.

[29] P. Sen, A. Deshpande, and L. Getoor. Exploiting shared correlations in probabilistic databases. *Proc. VLDB Endow.*, 1(1), 2008.

[30] M. A. Soliman, I. F. Ilyas, and K. C. Chang. Top-k query processing in uncertain databases. In *ICDE*, 2007.

[31] D. Z. Wang, E. Michelakis, M. N. Garofalakis, and J. M. Hellerstein. Bayesstore: managing large, uncertain data repositories with probabilistic graphical models. *PVLDB*, 1(1), 2008.

[32] K. Yi, F. Li, G. Kollios, and D. Srivastava. Efficient processing of top-$k$ queries in uncertain databases. In *ICDE*, 2008.

## Acknowledgments

Funding for this work was provided by Hong Kong RGC GRF Grant No. 611608 and NSFC Grant No. 60933011 and 60933012.

## Appendix

## A. Proof of Lemma 3.1.

**Proof.** It is sufficient to prove that for any uncertain object $o_i$ in node $e$, it always holds that $Pr_{LC\text{-}PNN}(q, o_i)$ in Inequality (1) equals to 0 (since $\alpha > 0$). Since $o_i \in e$ holds, we have $mindist(q, o_i) \geq mindist(q, e)$. Thus, by the assumption of lemma that $mindist(q, e) > best\_so\_far$ holds, we obtain $mindist(q, o_i) > best\_so\_far$ via inequality transition. As a result, the LC-PNN probability in Inequality (1) is always equal to 0, due to the existence of an object (we have seen so far) with maximum distance, $best\_so\_far$, to $q$ (i.e., $dist(q, \cdot) < r$ always holds for this object). Hence, the lemma holds. □

## B. Proof of Lemma 3.2.

**Proof.** Since the upper bound of $Pr_{LC\text{-}PNN}(q, o_i)$ is given by $UB\_Pr_{LC\text{-}PNN}(q, o_i)$, according to the lemma assumption that $UB\_Pr_{LC\text{-}PNN}(q, o_i) < \alpha$, we have $Pr_{LC\text{-}PNN}(q, o_i) < \alpha$. Thus, from Definition 2.1, object $o_i$ can be safely pruned. □

## C. Pseudo Code of LC-PNN Query Processing.

Figure 14 illustrates the pseudo-code of LC-PNN query processing, namely LC-PNN_Processing, which retrieves probabilistic nearest neighbor of a query point $q$ over uncertain data with local correlations, with probability greater than or equal to a threshold $\alpha \in (0, 1]$. Specifically, we conduct the LC-PNN query by traversing an R-tree index containing LCPs, as mentioned in our generic framework. To facilitate the query processing, we maintain a candidate set $S_{cand}$ (initially empty) for storing LC-PNN candidates, as well as a variable $best\_so\_far$ (initially $+\infty$) to keep the smallest distance upper bound from $q$ to objects we have seen so far (line 1). To traverse the index, we also create a *minimum heap* $\mathcal{H}$ accepting entries in the form $\langle e, key \rangle$ (line 2), where $e$ is either an R-tree node or a locally correlated partition (LCP), and $key$ is the sorting key in the heap defined as the minimum distance from query point $q$ to node/LCP $e$. Next, we insert the root, $root(\mathcal{I})$, of index $\mathcal{I}$ into heap $\mathcal{H}$ (line 3). Each time we pop out an entry $\langle e, key \rangle$ from the heap (line 5). If $key$ is greater than variable $best\_so\_far$, indicating the rest of entries in the heap cannot be closer to $q$ than that we have seen, we will terminate the loop (line 6); otherwise, we further check the children of $e$.

When $e$ is an LCP, we verify whether or not for each object $o_i \in e$, $mindist(q, o_i) \leq best\_so\_far$ holds. If the answer is yes, $o_i$ is a candidate and is added to the candidate set $S_{cand}$ lines 7-10); otherwise, $o_i$ can be discarded (via Lemma 3.1). In addition, we also load the pre-computed joint probabilities (e.g., $JP_o(\cdot)$), and and compute the probability upper bound, $UB\_Pr_{LC\text{-}PNN}(q, o_i)$. If it holds that $UB\_Pr_{LC\text{-}PNN}(q, o_i) < \alpha$, we can prune candidate $o_i$ by marking it as a false alarm.

On the other hand, no matter $e$ is a leaf or non-leaf node, for each child $e_i$ of $e$, we check whether or not $mindist(q, e_i) \leq best\_so\_far$ holds. If the answer is yes, then it indicates that $e_i$ may contain LC-PNN answers. Thus, we insert objects/nodes in $e_i$ (in the form $\langle e_i, mindist(q, e_i) \rangle$) into heap $\mathcal{H}$ for further filtering (lines 16-25). Further, we can apply the filtering method discussed in Section 4.1 via the pre-computed $JP_e(\cdot)$ (lines 20 and 25). The index traversal stops when either heap $\mathcal{H}$ is empty (line 4) or no nodes in the heap can contain LC-PNN answers (line 6).

After the index traversal, we can obtain and refine the remaining LC-PNN candidates by computing the actual LC-PNN probability (given in Inequality (1)) using the graphical model on locally correlated uncertain data (line 26). Finally, we return the actual LC-PNN answers satisfying Inequality (1) (line 27).

**Procedure** LC-PNN_Processing {
**Input:** an uncertain database $\mathcal{D}^U$ containing uncertain objects with local correlations, an R-tree index $\mathcal{I}$ over $\mathcal{D}^U$, a query point $q$, and a probabilistic threshold $\alpha \in (0, 1]$
**Output:** the answer to the LC-PNN query
(1) $S_{cand} = \phi$; $best\_so\_far = +\infty$;
(2) initialize an empty min-heap $\mathcal{H}$ accepting entries in the form $\langle e, key \rangle$
(3) insert $\langle root(\mathcal{I}), 0 \rangle$ into heap $\mathcal{H}$
(4) while heap $\mathcal{H}$ is not empty // Section 3.1
(5) $\langle e, key \rangle$ = de-heap ($\mathcal{H}$)
(6) if $key > best\_so\_far$, then break; // Lemma 3.1
(7) if $e$ is a locally correlated partition
(8) for each object $o_i \in e$
(9) if $mindist(q, o_i) \leq best\_so\_far$
(10) $S_{cand} = S_{cand} \cup \{o_i\}$
(11) load the pre-computed joint probabilities for object $o_i$ // Section 3.2
(12) if $UB\_Pr_{LC\text{-}PNN}(q, o_i) < \alpha$, then mark $o_i$ as false alarm
(13) if $best\_so\_far > maxdist(q, o_i)$
(14) let $best\_so\_far = maxdist(q, o_i)$
(15) filter out false alarms in $S_{cand}$ with the updated $best\_so\_far$
(16) if $e$ is a leaf node
(17) for each LCP $LCP_j \in e$
(18) if $mindist(q, LCP_j) \leq best\_so\_far$
(19) insert $\langle LCP_j, mindist(q, LCP_j) \rangle$ into heap $\mathcal{H}$
(20) filter candidates in $S_{cand}$ via $JP_e(\cdot)$ on $LCP_j$ // Eq. (7)
(21) if $e$ is an intermediate node
(22) for each entry $e_i \in e$
(23) if $mindist(q, e_i) \leq best\_so\_far$
(24) insert $\langle e_i, mindist(q, e_i) \rangle$ into heap $\mathcal{H}$
(25) filter candidates in $S_{cand}$ via $JP_e(\cdot)$ on node $e_i$ // Eq. (7)
(26) refine candidates in $S_{cand}$ by computing the actual LC-PNN probability
(27) return the final answer set
}

**Figure 14: Procedure of LC-PNN Query Processing**

## D. Discussions on Other Queries Over Locally Correlated Data.

We now discuss how to apply our filtering idea via pre-computations to answering other spatial queries over uncertain and locally correlated data in our framework (Section 2.2). In brief, our basic idea of filtering technique is to derive an upper bound of probability, that a candidate $o_i$ satisfies query predicates $\mathcal{P}$, by offline pre-computing joint probabilities in $LCS(o_i)$ and other LCPs via pivots (filtering in index nodes is similar to Lemma 4.1). One immediate query type is the LC-PNN query under metric distance functions (other than Euclidean distance in LC-PNN), such as $L_1$-norm. Since our pre-computations use pivots to derive probability upper bound, we can apply the similar technique as long as the underlying distance functions follow the triangle inequality.

Another interesting query is the dynamic skyline query [8], that retrieves uncertain objects that are not *dynamically dominated* by other objects (w.r.t. a query point $q$) with probability greater than or equal to a threshold. That is,

DEFINITION 8.1. (*Probabilistic Skyline Query on Uncertain Data with Local Correlations, LC-SKY*) *Given an uncertain database $\mathcal{D}^U$ containing locally correlated uncertain objects and a probabilistic threshold $\alpha \in (0, 1]$, an LC-SKY query retrieves those uncertain objects $o_i \in \mathcal{D}^U$ such that $o_i$ is not dominated by other objects with probability, $Pr_{LC\text{-}SKY}(o_i)$, greater than or equal to $\alpha$, that is,*

$$Pr_{LC\text{-}SKY}(o_i) = \int_{o_i \in UR(o_i)} \left( Pr\{o_i\} \cdot Pr\left\{ \bigwedge_{\forall u \in LCS(o_i)} u \not\prec o_i \mid o_i \right\} \cdot \left( Pr\left\{ \bigwedge_{\forall v \in \mathcal{D}^U \setminus (LCS(o_i) \cup \{o_i\})} v \not\prec o_i \right\} \right) \right) dr \quad (14)$$

*where $x \not\prec y$ indicates that $x$ is not dominated by $y$. In particular, we say that $x$ dominates $y$, if it holds that 1) $x[i] \leq y[i]$ for all dimensions $1 \leq i \leq d$; and 2) $x[j] < y[j]$ for at least one dimension $1 \leq j \leq d$.*

Similar to LC-PNN, we can pre-compute the joint probabilities that each sample of a candidate $o_i$ is not dynamically dominated w.r.t. pivots by objects in each of LCPs, considering different possible positions of $q$. This way, a probability upper bound can be derived from these pre-computations.

For top-$k$ queries [12], with a pre-selected preference function, we can also pre-compute joint probabilities that each sample of object $o_i$ has score higher than others in LCPs. We would like to incorporate other spatial queries on uncertain and locally correlated data into our framework as future work.

## E. Descriptions of Experimental Settings in Section 5.

We test our proposed approaches on both real and synthetic data sets. For the synthetic data, we generate the locally correlated partitions (LCPs), $LCP_j$, as follows. That is, we first pick up a center location, $C_{LCP_j}$, for LCP $LCP_j$ in a data space $\mathcal{U} = [0, 100]^d$. Then, we randomly produce the extent, $e_{LCP_j} \in [e^{LCP}_{min}, e^{LCP}_{max}]$, of $LCP_j$ on each dimension (i.e., along the $k$-th dimension, objects in $LCP_j$ are within $[C_{LCP_j}[k] - e_{LCP_j}[k], C_{LCP_j}[k] + e_{LCP_j}[k]]$). Within $LCP_j$, we further generate $n$ locally correlated uncertain objects on average. In particular, we randomly generate uncertain objects $o_i$ in $LCP_j$, each with object extent in $[e^{LCP}_{min}/2, e^{LCP}_{min}]$ on each dimension. We simulate the correlations among uncertain objects in each LCP $LCP_j$ as follows. We first create a large table having every coordinate of each object in $LCP_j$ as one attribute, then assign each tuple in the table with a joint probability (note: the summation of joint probabilities for all tuples in the table is equal to 1), and finally construct conditional probability tables (CPTs) from this large table based on a probabilistic graphical model (graph structure is randomly generated, i.e., producing random edges between pairs of objects until the graph is connected, like the one in Figure 2(b)). We denote the data set with LCP center *location* $C_{LCP_j}$ following *Uniform* (or *Skew* with skewness 0.8) distribution as $lU$ ($lS$); similarly, denote the data set with LCP *extent* following *Uniform* (*Gaussian* with mean $(e^{LCP}_{min} + e^{LCP}_{max})/2$ and variance $(e^{LCP}_{max} - e^{LCP}_{min})/5$) distribution as $eU$ ($eG$). Thus, we have four types of data sets, $lUeU$, $lUeG$, $lSeU$, and $lSeG$. In our experiments, we set the attribute domain of objects in each LCP to 10 and the default dimensionality to 2. Note that, for other data sets with different parameters or distributions (e.g., dimensionality, mean, variance, skewness), the query trends are similar.

Moreover, for the real data, we use a spatial data set, $CA$, which contains nodes of California Road Network obtained from Digital Chart of the World Server [*http://www.maproom.psu.edu/dcw/*]. We consider each data point in $CA$ as the center $C_{LCP_j}$ of an LCP, and randomly generate its extent $e_{LCP_j}$ following either Uniform or Gaussian distribution (denoted as $CA\_eU$ and $CA\_eG$, respectively). The resulting data sets can simulate the scenarios with locally correlated spatial objects. For each of the data sets above, we insert LCPs into an R*-tree index [1] with page size 4K, on which the LC-PNN query is processed. To enable the candidate filtering, we offline pre-compute the joint probabilities (by using 10 selected $\lambda$ values within $[min\_\lambda, max\_\lambda]$ in our experiments), as mentioned in Section 3.2.