Probabilistic Maximum Range-Sum Queries on Spatial Database (technical report)

Qiyu Liu, Xiang Lian and Lei Chen
Probabilistic Maximum Range-Sum Queries on Spatial Database (Technical Report)

Qiyu Liu
HKUST
Hong Kong SAR
qliuau@cse.ust.hk

Xiang Lian
Kent State University
Kent, USA
xlian@kent.edu

Lei Chen
HKUST
Hong Kong SAR
leichen@cse.ust.hk

ABSTRACT
Maximum Range-Sum (MaxRS) query is an important operator in spatial database for retrieving regions of interest (ROIs). Given a rectangular query size $a \times b$ and a set of spatial objects associated with positive weights, MaxRS retrieves rectangular regions $Q$ of size $a \times b$, such that the sum of object weights covered by $Q$ (i.e., range-sum) is maximized. Due to the inaccuracy of the location acquisition, the collected locations of spatial objects are inherently uncertain and imprecise, which can be modeled by uncertain objects. In this paper, we propose a Probabilistic Maximum Range-Sum (PMaxRS) query over uncertain spatial objects, which obtains a set $\gamma^*$ of rectangles such that the probability that each region $Q \in \gamma^*$ has the maximum range-sum exceeds a user-specified threshold $P_t$. We show that determining whether a given region $Q$ is in the answer of PMaxRS query is as hard as #$P$-complete. To solve that, we put forward an efficient $P_{MaxRS}$ Framework based on pruning and refinement strategies. In the pruning step, we propose a candidate generation technique to reduce the search space. In the refinement step, we design an efficient sampling-based approximation algorithm to verify the remaining candidate regions. Extensive experiments are conducted to demonstrate the effectiveness and efficiency of our algorithms.

KEYWORDS
Probabilistic Maximum Range-Sum Query, Approximate Algorithm, Uncertain Database

1 INTRODUCTION
With the prevalence of GPS-enabled mobile devices and the popularity of location-based services, spatial data management has become increasingly important for providing location-based services. As a fundamental operator of spatial database, optimal location queries [15, 31, 33–35, 37] have been well studied by the database community, such as optimal location selection [15, 34], bichromatic reverse nearest neighbor queries [31, 37] and top-k spatial queries [33, 35]. Different from these queries which rank and select spatial objects from a candidate set, a Maximum Range-Sum (MaxRS) query [5, 13, 18, 24, 29] retrieves a region with a user-specified size covering sites that users are most likely interested in. Formally, a MaxRS query retrieves a $a \times b$ rectangular regions $Q$ that contain objects with the highest sums of their weights (called range-sum).

Due to the usefulness in retrieving regions of interest (ROIs), MaxRS queries over certain data have attracted a lot of attention recently [5, 6, 10, 13, 23, 29]. However, in many real applications, the data uncertainty exists in spatial database naturally due to various factors such as privacy issues, data incompleteness, GPS device inaccuracy or network transmission errors. As a consequence, it is not trivial to obtain accurate MaxRS answers over uncertain spatial databases, which, to the best of our knowledge, has not been investigated so far. Below are two examples showing that answering MaxRS queries under uncertain semantics is useful in real applications.

Example 1. (Discovering Future Traffic Jam). Given historical traffic data of a city, some trajectory mining algorithms like [22] can discover the trajectory patterns and use them to predict the next location of a car in the future. Such prediction algorithms usually output a predicted location associated with a confidence value (i.e., the probability). All the predicted locations and their corresponding probabilities form an uncertain database. By invoking a MaxRS query over such an uncertain database, we can discover possible jam regions with high probabilities at future timestamps, which is the central issue for urban transportation management.

Example 2. (Privacy-preserving Crowd Detection). Many applications such as location-aware advertising, route planning and spatial crowdsourcing [1, 2, 9] require the location information of possible crowds. Thus, crowd detection, which discovers the densely distributed users or workers, is important in location-based services. However, due to the consideration of protecting user’s privacy, instead of concrete location of a user, we get the distribution of each user’s location which has been disturbed before sending to the server by some specific privacy-preserving algorithms like [30]. Figure 1 illustrates the scenario in Example 2. $U_1$, $U_2$ and $U_3$ are three users whose location information has been disturbed due to privacy issues. $u_{ij}$ denotes the possible position of user $U_i$. On such uncertain data, a MaxRS query retrieves the possible locations of crowds with high confidence for location-based services like advertisement recommendation and task assignment in spatial crowdsourcing.

Note that, Example 2 is a special case where all the weights are set to 1. The term weight used in this paper could be defined with different semantics which is related to different applications, such as the score of a restaurant, the potential of customer or simply 1 so that range-sum reflects the COUNT of objects in a region.
To support MaxRS over an uncertain database, we propose a Probabilistic Max Range-Sum (PMaxRS) query on uncertain databases. Under the uncertain setting, each spatial object has multiple possible instances associated with different positions, weights and probabilities. The inputs of a PMaxRS query are: (i) a set of uncertain objects \( \mathcal{U} \), each object associated with a weight and a probability; (ii) the query size \( a \times b \) of rectangular regions, and (iii) a user-specified probabilistic threshold \( P_t \). The PMaxRS output is the position of a rectangle \( Q \), such that the probability that \( Q \) has the maximum range-sum (namely, the PMaxRS probability) exceeds threshold \( P_t \).

**Challenges.** To the best of our knowledge, no prior research has studied the MaxRS problem with the location uncertainty, and current solutions to the MaxRS problem over certain databases [13, 15, 29] cannot be easily applied to the uncertain scenario. Besides, like many other queries on uncertain databases [8, 19, 25, 27, 36], the PMaxRS problem suffers from the exponential cardinality of possible worlds [3] over uncertain data (i.e., \( O(K^N) \), where \( N \) is the number of objects and \( K \) is the average number of instances per uncertain object). Theoretically, we will show that verifying whether a given region is the PMaxRS solution, a special case of the PMaxRS problem, is as hard as the \#P-complete \([16]\) problem, which has been shown to be \#P-complete \([16]\) [16]. This inspires us to design approximation algorithms to speed up the PMaxRS query processing with small accuracy loss.

**Contributions.** We list the major contributions as follows:

- We formulate the PMaxRS problem in the context of uncertain databases by using possible worlds semantics. To answer PMaxRS queries efficiently, we propose a two-phase algorithm, called PMaxRS_Framework, based on pruning and refinement strategies.
- We derive an upper bound of the PMaxRS probability, based on which we design candidate selection rules to significantly reduce the search space.
- Due to the \#P-hardness of the verification whether a candidate region is the PMaxRS solution, we design a sampling-based approximate refinement algorithm. We prove that by an appropriate selection of the sample size, the sample error can be arbitrarily small with high probability.
- We conduct extensive experiments using several datasets with different parameter settings to demonstrate the effectiveness and efficiency of our proposed algorithms.
- We show that our PMaxRS_Framework can be easily extended to answer PMaxRS queries of circular case.

**Organization.** The rest of our paper is organized as follows. In Section 2, we formally define the PMaxRS problem and introduce the PMaxRS_Framework. In Sections 3 and 4, we discuss details of each component in PMaxRS_Framework, including the pruning bound, space traversing algorithm, and refinement techniques. In Section 5, we present our experimental results and analyze the effects of various parameter settings. Previous works on the MaxRS problem and uncertain data management are reviewed and compared with our work in Section 6. Finally, in Section 7, we conclude this paper.

## 2 PROBLEM DEFINITION

In this section, we formulate the PMaxRS problem by using the possible worlds semantics [3]. For quick reference, all the notations and symbols used in this paper can be found in Table 1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{U} )</td>
<td>an uncertain database</td>
</tr>
<tr>
<td>( N )</td>
<td>the number of uncertain objects</td>
</tr>
<tr>
<td>( U_i )</td>
<td>an uncertain object</td>
</tr>
<tr>
<td>( u_{i1}, \ldots, u_{iK} )</td>
<td>( K ) instances of an uncertain object ( U_i )</td>
</tr>
<tr>
<td>( PW )</td>
<td>the space of all possible worlds</td>
</tr>
<tr>
<td>( pw )</td>
<td>a possible world</td>
</tr>
<tr>
<td>( \mathcal{Q}_x )</td>
<td>a rectangular region centered at ( x )</td>
</tr>
<tr>
<td>( Q_{x, \text{sum}(pw)} )</td>
<td>range-sum of ( \mathcal{Q}_x ) over possible world ( pw )</td>
</tr>
<tr>
<td>( P_{\text{max}}(\mathcal{Q}_x) )</td>
<td>PMaxRS probability of given region ( \mathcal{Q}_x )</td>
</tr>
</tbody>
</table>

### 2.1 Preliminaries

We first formally define the MaxRS problem over a certain database.

**Definition 1.** (MaxRS). Given a user-specified \( a \times b \) rectangular query region and a set of spatial objects \( O = \{ o_1, o_2, \ldots, o_N \} \), each object \( o_i \in O \) associated with a non-negative weight \( w_i \), and a location \( l_i \in \mathbb{R}^2 \), the MaxRS problem aims at retrieving an optimal rectangular region \( \mathcal{Q}_x^* \) centered at location \( x \) and with size \( a \times b \) such that:

\[
\mathcal{Q}_x^* = \arg\max_{x \in \mathbb{R}^2} \sum_{l_i \in \mathcal{Q}_x} w_i, \tag{1}
\]

where \( l_i \in \mathcal{Q}_x \) is the shorthand of \( l_i \) falls into the region \( \mathcal{Q}_x \).

Note that, we adopt the rectangular shape of region \( \mathcal{Q}_x \) following previous researches on the MaxRS problem [5, 13, 18, 24, 29]. However, we find that all of our theoretical results can be naturally extended to the circular cases. Unless otherwise specified, the term “region” refers to the rectangular region hereafter.

### 2.2 Probabilistic Data Model

**Definition 2.** (Uncertain Spatial Object). An uncertain spatial object is defined as \( U_i = \{ u_{i1}, u_{i2}, \ldots, u_{iK} \} \), where \( u_{i1}, \ldots, u_{iK} \) are \( K \) possible instances of \( U_i \). Each instance \( u_{ij} \in U_i \) is associated with a non-negative weight \( w_{ij} \), an existence probability \( p_{ij} \), and a spatial position \( l_{ij} \), with constraint \( \sum_{j=1}^{K} p_{ij} = 1 \).

Let \( \mathcal{U} = \{ U_1, U_2, \ldots, U_N \} \) be a collection of uncertain spatial objects, i.e., an uncertain spatial database.

**Definition 3.** (Possible World). A possible world \( pw = \{ u_{ij} | i = 1, 2, \ldots, N \} \) is defined as a materialized instance of \( \mathcal{U} \) where each uncertain object \( U_i \) takes an instance \( u_{ij} \). The appearance probability of a possible world \( pw \) is \( Pr(pw) = \prod_{i=1}^{N} p_{ij} \). Let \( PW \) denote the whole space of \( pw \), i.e., \( PW = U_1 \times U_2 \times \cdots \times U_N \) where \( \sum_{pw \in PW} Pr(pw) = 1 \).

For the uncertain spatial objects shown in Figure 1, the corresponding weights and probabilities of all instances are shown in Table 2. On such an uncertain database, a possible world could be \( pw = \{ u_{11}, u_{22}, u_{31} \} \) and its probability is \( 0.3 \times 0.5 \times 0.8 = 0.12 \).

Our definition and solution to the discrete case could be easily extended to the continuous case where each uncertain object is associated with a continuous density function by using re-sample technique like bootstrap. Due to the space limitation, we only discuss the discrete case.

By following the convention [21, 25, 27], we assume that uncertain objects in the database are independent of each other.
2.3 The PMaxRS Problem

Different from MaxRS problem over an exact database, for a given rectangular region \(Q_x\) with size \(a \times b\) centered at \(x\), we consider the probability that \(Q_x\) is the MaxRS solution, which is called the “PMaxRS Probability”. By using the possible worlds semantics, the PMaxRS probability can be defined as follows.

**Definition 4. (The PMaxRS Probability).** For a given region \(Q_x\), its PMaxRS probability over an uncertain spatial database, denoted by \(Pr_{\text{maxrs}}(Q_x)\), is defined as follows,

\[
Pr_{\text{maxrs}}(Q_x) = \sum_{pw \in PW} \delta(Q_x||pw) \cdot Pr(pw),
\]

where \(\delta(Q_x||pw)\) is an indicator function:

\[
\delta(Q_x||pw) = \begin{cases} 1, & \text{if } Q_x.\text{sum}(pw) = \text{max } Q_x'.\text{sum}(pw); \\ 0, & \text{otherwise}, \end{cases}
\]

where \(Q_x.\text{sum}(pw)\) denotes the range-sum of region \(Q_x\) on possible world \(pw\), i.e.,

\[
Q_x.\text{sum}(pw) = \sum_{\forall w_{ij} \in pw, u_{ij} \in Q_x} w_{ij}. \tag{3}
\]

According to Definition 4, given a possible world (actually, can be regarded as a certain database) \(pw \in PW\) and a region \(Q_x\), the indicator function \(\delta(Q_x||pw)\) denotes whether region \(Q_x\) covers the highest range-sum over possible world \(pw\) (we call \(Q_x\) is optimal on \(pw\) if \(\delta(Q_x||pw) = 1\)). Thus, \(\delta(Q_x||pw) \cdot Pr(pw)\) is the probability that region \(Q_x\) is in the answer of the exact MaxRS query on possible world \(pw\). Then, \(Pr_{\text{maxrs}}(Q_x)\) can be interpreted as the probability that \(Q_x\) is optimal among all possible worlds \(pw \in PW\).

In the previous example of Figure 1 and Table 2, we consider PMaxRS probabilities of 5 candidate rectangular regions \(Q_1\) to \(Q_5\), whose locations are shown in Figure 1. Table 3 lists possible world(s) and PMaxRS probabilities of \(Q_1\) (i.e., \(Pr_{\text{maxrs}}(Q_1)\)). As an example, \(Q_1\) is optimal on possible worlds \{\(u_{11}, u_{21}, u_{31}\), \(u_{11}, u_{22}, u_{31}\), \(u_{12}, u_{22}, u_{31}\) and \(u_{13}, u_{22}, u_{31}\}\), and its corresponding PMaxRS probability is the sum of probabilities of these 4 possible worlds, which is 0.52.

With the definition of the PMaxRS probability, a PMaxRS query could be defined as an operator over an uncertain spatial database which retrieves all the regions with PMaxRS probabilities larger than a user-specified threshold.

**Definition 5. (The PMaxRS Query).** Given a set of uncertain objects \(\mathcal{U} = \{U_1, U_2, \cdots, U_N\}\), the size of query region \((a, b)\), and a probability threshold \(P_t\), a PMaxRS query retrieves a set of regions \(\gamma^*\) with size \(a \times b\) such that for \(\forall Q_x \in \gamma^*\), \(Pr_{\text{maxrs}}(Q_x) \geq P_t\); namely, the result of a PMaxRS query is:

\[
\gamma^* = \{Q_x | Pr_{\text{maxrs}}(Q_x) \geq P_t\}. \tag{4}
\]

For the example discussed above, if the probabilistic threshold \(P_t\) is specified as 0.5, then \(Q_4\) is in the PMaxRS solution set. Similarly, if \(P_t = 0.3\), then both \(Q_2\) and \(Q_4\) are PMaxRS query answers.

**Hardness.** Intuitively, answering PMaxRS query is exponentially hard. According to Definition 5, for arbitrary region \(Q_x\), direct calculation of the PMaxRS probability, \(Pr_{\text{maxrs}}(Q_x)\), requires enumerating all possible worlds, which yields \(\prod_{i=1}^{N} |U_i| = O(K^N)\) time complexity. To theoretically demonstrate the hardness of the PMaxRS query, we show that, the decision version of the PMaxRS problem, i.e., determining whether a given region \(Q_x\) is in the answer set of the PMaxRS query over an uncertain database, is already as hard as the #KNAPSACK problem, which is the counting version of the classic 0-1 KNAPSACK problem and has already been shown as \#P-complete [16]. Theorem 1 reveals the hardness as follows.

**Theorem 1. (PMaxRS Hardness).** Given an uncertain database \(\mathcal{U}\) and a rectangular region \(Q_x\), the verification of whether the PMaxRS probability \(Pr_{\text{maxrs}}(Q_x)\) exceeds the user-specified probabilistic threshold \(P_t\) is as hard as #KNAPSACK, which is \#P-complete.

### 2.4 Solution Overview

To handle the exponential hardness of answering PMaxRS queries, we develop an efficient PMaxRS Framework based on pruning and refinement strategy as shown in Algorithm 1. In line 1, a subroutine \textsc{CandidateGen}, introduced in Section 3, traverses all possible candidate regions and filters out the ones whose PMaxRS probabilities are less than threshold \(P_t\) (i.e., impossible to be PMaxRS solution). To support that, we analyze the upper bound of \(Pr_{\text{maxrs}}(Q_x)\) denoted as \(ub_P(Q_x)\). For an arbitrary region \(Q_x\), if \(ub_P(Q_x) < P_t\), it is impossible for \(Q_x\) to satisfy the probabilistic constraint, and thus \(Q_x\) can be safely pruned. To efficiently generate candidates, we can maintain a sweep-line to traverse the whole space. With the candidate set returned by \textsc{CandidateGen}, denoted by \(C\), in the refinement step shown in line 2, for each candidate region \(Q \in C\), a subroutine \textsc{Verify}, shown in Section 4, checks whether it is indeed in PMaxRS solution set.

**Algorithm 1: PMaxRS Framework**

1. **Input:** an uncertain database \(\mathcal{U}\), query size \(a \times b\) and probability threshold \(P_t\)
2. **Output:** PMaxRS solution set \(\gamma^*\)
   - \(C \leftarrow \text{CandidateGen}(\mathcal{U}, (a, b), P_t)\); // The Pruning Step
   - \(y^* \leftarrow \text{Verify}(C, \mathcal{U}, P_t);\) // The Refinement Step
   - return \(y^*\);

### 3 THE PRUNING STEP

In this section, we introduce the pruning heuristics and design an efficient traversing algorithm for finding all candidate regions in the whole space. We first reduce \(Pr_{\text{maxrs}}(Q_x)\) shown in Definition 5 over possible worlds to another formula w.r.t. uncertain objects in probabilistic spatial database. Then, we develop the upper bound of
the PMaxRS probability of a given region \(Q_x\), and derive pruning conditions for candidate regions.

### 3.1 Problem Reduction

The intuition we do the problem reduction is that, instead of directly computing \(Pr_{\text{maxrs}}(Q_x)\) as Eq. (2), we aim to represent the PMaxRS probability via statistics like mean and variance, and based on which we derive the probabilistic bound. We first define the range-sum over probabilistic spatial database.

**Definition 6.** (Range-Sum on Probabilistic Spatial Database).

Given a region \(Q_x\), its range-sum over a probabilistic spatial database \(\mathcal{U}\), denoted by \(Q_x.\sum\), is defined as a random variable which takes values as all possible sums of weights of instances falling into \(Q_x\).

To illustrate \(Q_x.\sum\), we consider the region \(Q_4\) in Figure 1 as an example, where \(Q_4\) covers two uncertain objects \(U_2\) and \(U_3\). Specifically, \(Q_4\) covers instances \(u_{22}\) and \(u_{24}\). We enumerate all possibilities of \(Q_4.\sum\) in Table 4. There are 4 possible values of \(Q_4.\sum\). For example, if uncertain object \(U_2\) takes instance \(u_{22}\) and \(U_3\) takes \(u_{31}\), then \(Q_4.\sum\) takes value \(w_{22} + w_{31} = 1.3\) and its corresponding probability is \(p_{22} × p_{31} = 0.4\). Similarly, if \(U_2\) takes \(u_{22}\) and \(U_3\) takes \(u_{32}\) (note that, \(u_{32}\) is outside of \(Q_4\) which means it contributes 0 to \(Q_4.\sum\)), then \(Q_4.\sum\) takes value \(w_{22} + 0 = 0.4\) and its corresponding probability is \(p_{22} × p_{32} = 0.1\).

**Table 4: Possible range-sums of \(Q_4\) in Figure 1.**

<table>
<thead>
<tr>
<th>(Q_4.\sum)</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w_{22} + w_{31} = 1.3)</td>
<td>(p_{22} × p_{31} = 0.4)</td>
</tr>
<tr>
<td>(w_{22} + 0 = 0.4)</td>
<td>(p_{22} × p_{32} = 0.1)</td>
</tr>
<tr>
<td>(0 + w_{31} = 0.9)</td>
<td>(p_{21} × p_{31} = 0.4)</td>
</tr>
<tr>
<td>(0 + 0 = 0)</td>
<td>(p_{21} × p_{32} = 0.1)</td>
</tr>
</tbody>
</table>

We then have the PMaxRS probability reduction lemma:

**Lemma 1.** (PMaxRS Reduction). Given a region \(Q_x\), it always holds that \(Pr_{\text{maxrs}}(Q_x) = Pr(\bigwedge_{x' \in x} Q_{x'}.\sum \geq Q_{x}.\sum)\) where \(x' \in \mathbb{R}^2\), and \(Q_{x'}\) is an a \& b rectangular region centered at \(x'\), and \(\forall x'\) traverses the entire \(\mathbb{R}^2\) space.

Lemma 1 transforms the PMaxRS probability, \(Pr_{\text{maxrs}}(Q_x)\), defined over possible worlds to \(Pr(\bigwedge_{x' \in x} Q_{x'}.\sum \geq Q_{x}.\sum)\), which is described by uncertain objects and instances. By Lemma 1, we transform the calculation of \(Pr_{\text{maxrs}}(Q_x)\) to an equivalent way which is analysis-friendly, and based on which, we derive the upper bound of \(Pr_{\text{maxrs}}(Q_x)\) and corresponding pruning conditions in the following subsections.

### 3.2 Pruning Bound

To derive the upper bound of \(Pr_{\text{maxrs}}(Q_x)\), we select a relatively “good” region \(Q_{\mathcal{A}}\) as a reference region. The intuition is that, we can use \(Q_{\mathcal{A}}\) to prune other candidate regions \(Q_x\) which are much worse than \(Q_{\mathcal{A}}\). Based on this idea, we derive an upper bound of \(Pr_{\text{maxrs}}(Q_x)\):

**Lemma 2.** (Probability Upper Bound). Given an arbitrary region \(Q_x\) and a reference region \(Q_{\mathcal{A}}\), it holds that:

\[
Pr_{\text{maxrs}}(Q_x) \leq \begin{cases} \frac{\mu^2}{\sigma^2 + \mu^2}, & \text{if } E[Q_x.\sum] < E[Q_{\mathcal{A}}.\sum] \\ 1, & \text{otherwise.} \end{cases}
\]

where \(\mu = E[Q_x.\sum - Q_{\mathcal{A}}.\sum]\) and \(\sigma^2 = \text{Var}[Q_x.\sum - Q_{\mathcal{A}}.\sum]\).

Lemma 2 provides an upper bound of \(Pr_{\text{maxrs}}(Q_x)\). Recall that, the PMaxRS query retrieves \(Q_x\) with \(Pr_{\text{maxrs}}(Q_x)\) greater than probabilistic threshold \(P_t\) (as given in Definition 5). The basic idea of our pruning method is to safely filter out those regions \(Q_x\) with probability upper bound no less than the probability threshold \(P_t\).

Theoretically, the reference region \(Q_{\mathcal{A}}\) can be set as an arbitrary region, and the upper bound derived in Lemma 2 always holds. However, the selection of \(Q_{\mathcal{A}}\) influences the pruning power. Intuitively, a “good” reference region should have high expectation and low variance, and thus can decrease the upper bound of \(Pr_{\text{maxrs}}(Q_x)\), which improves the pruning power. However, finding an optimal reference region for all possible \(Q_x\) is quite difficult. The following heuristic strategies can be adopted for the selection:

(i) MaxPossible: let \(Q_{\mathcal{A}}\) be the MaxRS solution over the possible world with the highest probability;

(ii) k-Sampling: sample \(k\) possible worlds and select \(Q_{\mathcal{A}}\) as the region with the highest frequency to be MaxRS solution over \(k\) possible worlds;

(iii) EMaxRS: let \(Q_{\mathcal{A}}\) be the expected MaxRS solution over all possible worlds.

In Section 5, we report the pruning power of different selection strategies and discuss how to choose \(Q_{\mathcal{A}}\) in practice to make a good trade-off between pruning time and pruning power.

### 3.3 Candidate Region Selection Rules

In this subsection, we introduce how to derive candidate selection criterion based on the reverse of the pruning bound discussed in Section 3.2. The basic idea is that, for a given region \(Q_x\), according to Eq. (13), we calculate the upper bound of \(Pr_{\text{maxrs}}(Q_x)\), denoted as \(ub(Pr_{\text{maxrs}}(Q_x))\). If \(ub(Pr_{\text{maxrs}}(Q_x))\) is larger than the user-specified threshold \(P_t\), \(Q_x\) is selected as a candidate and waits for further verification. Otherwise, \(Q_x\) can be safely filtered. Since specific candidate selection rules depend on whether \(Q_x.\sum\) and \(Q_{\mathcal{A}}.\sum\) are correlated, we discuss Independent Case and Correlated Case, respectively.

#### 3.3.1 Independent Case

Assume that \(Q_x.\sum\) and \(Q_{\mathcal{A}}.\sum\) are independent (i.e., \(Q_x\) and \(Q_{\mathcal{A}}\) do not contain instances from the same uncertain objects and Figure 2 illustrates such case). For brevity, we use \(\mu_{Q_x}, \sigma^2_{Q_x}\) and \(\sigma^2_{Q_{\mathcal{A}}}\) to denote \(E[Q_x.\sum], E[Q_{\mathcal{A}}.\sum], \text{Var}[Q_x.\sum]\) and \(\text{Var}[Q_{\mathcal{A}}.\sum]\), respectively. Then, we have the following candidate selection lemma for the independent case.

**Lemma 3.** (Independent Candidate Selection). Given a reference region \(Q_{\mathcal{A}}\) and an arbitrary region \(Q_x\), if one of the following conditions is satisfied, \(Q_x\) is selected as a candidate region.

\[
\begin{align*}
(i) \quad & \mu_{Q_x} \geq \mu_{Q_{\mathcal{A}}} \\
(ii) \quad & M_S > P_t \cdot \mu_{Q_{\mathcal{A}}} - \sigma^2_{Q_{\mathcal{A}}} \quad \text{and} \\
&m_{Q_x} \in \left[\min \left\{0, \frac{2P_t \cdot \sigma^2_{Q_x} - (1 - P_t) \Delta}{\sigma^2 + \mu^2}_{Q_x}\right\} \right] \\
&\Delta = \frac{1}{1 - P_t} \left(\text{MS} + \sigma^2_{Q_x} - P_t \cdot \mu^2_{Q_{\mathcal{A}}}\right).
\end{align*}
\]

where \(\Delta\) is the mean square of \(Q_x.\sum\), that is, \(\sigma^2_{Q_x} + \mu^2_{Q_{\mathcal{A}}}\), and \(\Delta\) is a function of \(P_t\).
3.3.2 Correlated Case. Now we discuss the correlated case where \( Q_x \) and \( Q_A \) are correlated. The reason leading to correlation is that \( Q_x \) and \( Q_A \) have mutually exclusive instances from the same uncertain objects. Figure 3 illustrates this case, where uncertain object \( U_2 \) intersects both \( Q_x \) and \( Q_A \). If \( U_2 \) takes instance \( u_2^1 \), it is impossible that \( u_2^3 \) appears in \( Q_A \), and vice versa.

The correlation leads to significant increase of the complexity, since we cannot decompose \( \text{Var}(Q_x \text{sum} - Q_A \text{sum}) \) and need to enumerate all possible combinations of shared instances. When the correlation happens, i.e., \( Q_x \) and \( Q_A \) share common uncertain objects, one straightforward way to reduce the computation is to skip this region and set \( Q_A \) as a candidate region. However, such an approach apparently decreases the pruning power. To make a trade-off between the pruning power and computational complexity, we perform an intuitive transformation which turns the correlated case into the independent case and then apply the candidate selection rule discussed in Section 3.3.1.

3.4 Candidate Generation Algorithm

In this subsection, we focus on how to traverse the data space and apply the candidate selection rules discussed above to find out all PMaxRS candidate regions. Note that, possible candidate region \( Q_x \) may locate at everywhere in \( \mathbb{R}^2 \) space. Thus, the question is, given an uncertain spatial database \( \mathcal{U} \), is the set of all possible candidate regions finite? We show that the answer is positive in Lemma 4.

**Lemma 4. (Number of All Candidates).** Given an uncertain database \( \mathcal{U} \), the lower and upper bounds of the total number of all possible candidate regions are \( \Omega(NK) \) and \( O(NK^2) \), respectively, where \( N \) is the total number of uncertain objects and \( K \) is the average number of instances per uncertain object.

**Proof.** We prove the lemma by transforming the original probabilistic spatial database to a set of rectangles. For each instance \( u_{ij} \), we draw a rectangle of size \( a \times b \) centered at \( l_{ij} \). The corresponding rectangle of instance \( u_{ij} \) is denoted by \( r_{ij} \). Figure 5(a) shows an example database with three uncertain objects \( U_1, U_2 \) and \( U_3 \). Figure 5(b) presents the transformation result. After the transformation, each overlapping area represents one possible candidate. For example, for any location \( x \) falling into the shaded area (i.e., the overlapping area of \( r_{11}, r_{12}, \) and \( r_{13} \)) in Figure 5(b), \( Q_x \) covers three instances: \( u_{11}, u_{12}, \) and \( u_{13} \). Thus, the total number of all possible candidate regions is equal to the total number of different overlapping areas. Then, it is not difficult to find that, the total number of overlapping areas is between \( NK = \Omega(NK) \) and \( NK(NK-1) = O(N^2K^2) \) regardless of the distribution of data points. Hence, the lemma holds.

**Figure 2:** Example of case that \( Q_x \) and \( Q_A \) are independent.

**Figure 3:** Example of case that \( Q_x \) and \( Q_A \) are correlated.

**Figure 4:** Transformation from correlated case to independent case.

**Figure 5:** Transformation to rectangle intersection problem.
With all the details discussed above, we give the implementation of the candidate generation technique called CANDIDATE_GEN in Algorithm 2. Line 1 transforms the input probabilistic spatial database into a set of rectangles denoted by \( \mathcal{R} \). In lines 3-6, a sweep-line \( \ell \) scans from left to right. At each state of \( \ell \), we invoke IS_CANDIDATE on the possible candidate regions generated by overlapping areas swept by \( \ell \) and append the region which satisfies the candidate selection rules into \( C \). The details of subroutine IS_CANDIDATE are shown in lines 8-12. Line 9 checks whether \( Q_x \), \( \sum \), and \( Q_r \), \( \sum \) are correlated. If yes, we transform \( Q_r \) to \( Q_r' \) which shares no common uncertain objects with \( Q_x \). Then, we invoke subroutine INDEPENDENT_PRUNING, shown in lines 5-9, to determine whether region \( Q_x \) is in PMaxRS candidate set by using the candidate selection rules discussed in Lemma 3.

**Algorithm 2: CANDIDATE_GEN**

**Input:** an uncertain database \( \mathcal{U} \), a query size \( a \times b \), a reference region \( Q_r \), and a probability threshold \( P_t \)

**Output:** a set of candidate regions \( C \)

1. \( \mathcal{R} \leftarrow \text{TRANSFORM}(\mathcal{U}, a, b) \)
2. \( C \leftarrow \phi \)
3. while sweeping the vertical line \( \ell \) from left to right on \( \mathcal{R} \) do
   4. for each \( Q_x \) generated by overlapping areas swept by \( \ell \) do
      5. if IS_CANDIDATE(Q, \( Q_r \), \( P_t \)) then
         6. \( C \leftarrow C \cup \{Q\} \)
5. return \( C \);

**Function IS_CANDIDATE**

1. if \( Q_x \) and \( Q_r \) share common uncertain objects then
   2. \( Q_r' \leftarrow \text{transformed region of} \ Q_r \)
   3. return INDEPENDENT_PRUNING(Q_x, \( Q_r' \), \( P_t \))
4. else return INDEPENDENT_PRUNING(Q_x, \( Q_r \), \( P_t \))

**Function INDEPENDENT_PRUNING**

1. calculate MS, \( \mu^{Q_x} \), \( \sigma^2_{Q_x} \), and \( \mu_r \)
2. if \( \mu^{Q_x} > \mu^{Q_r} \) then
   3. return True.
4. if MS > \( P_t \cdot \mu^{Q_x} - \sigma^2_{Q_x} \) and \( \mu_r \in \max(\frac{2P_t \mu^{Q_r} - (1-P_t)\mu_r}{\sigma_{Q_r}}, \min(\frac{2P_t \mu^{Q_r} - (1-P_t)\mu_r}{\sigma_{Q_x}})) \) then
   5. return True.
6. return False.

Note that, there are several tricks that can accelerate CANDIDATE_GEN. First, candidate regions can be generated from existing candidate set; that is to say, an Apriori-like [4] pruning strategy can be used to reduce total candidate generation time. Besides, since subroutine IS_CANDIDATE is frequently invoked, for efficient calculation of MS and \( \mu_r \), we can borrow the idea of generating function [20] which is first used for evaluating rank distribution over a probabilistic database. Due to the space limitation, we just give a simple example to show how generating function works in our problem and for the details of generating function and unified ranking techniques in uncertain database, the author can refer to work of Li et al. [20]. Take region \( Q_x \) shown in Figure 1 as an example, we can write down the generating function of \( Q_x \) sum as \( F_{Q_x} = (p_{22} \cdot x^{3} + (1 - p_{22}) \cdot x^3) \times (p_{31} \cdot x^{2} + (1 - p_{31}) \cdot x^2) \times (0.5x^{0.5} + 0.5x^{0.5}) \times (0.8x^{0.8} + 0.2x^{0.2}) \). Expand \( F_{Q_x} \), we have, \( 0.4x^{1.3} + 0.4x^{0.9} + 0.1x^{1.0} + 0.1x^{0} \). The monomial \( 0.4x^{1.3} \) can be interpreted as \( \Pr (Q_x, \sum = 1.3) = 0.4 \). As the sweep-line scanning from left to right, we dynamically update and expand the generating function, which will accelerate the evaluation of distribution of \( Q_x \) sum.

**Complexity Analysis.** We analyze the worst case time complexity of Algorithm 2. First, transforming the probabilistic spatial database \( \mathcal{U} \) to a set of rectangles takes \( O(NK) \) time, and sorting them by x-coordinate takes \( O(NK \log(NK)) \). Then, according to Lemma 4, there are at most \( O(N^2K^2) \) possible candidate regions. By adopting the generating function technique to calculate MS and \( \mu^{Q_x} \), the total time of updating and expanding generating functions can be bounded within \( O(N^2K^2) \) according to [20]. Thus, the total time complexity of Algorithm 2 in the worst case is \( O(N^2K^2) \).

**4 THE REFINEMENT STEP**

In this section, given a set of candidate regions generated by Algorithm 2, we consider how to verify whether these regions are truly in the solution set (i.e., determining whether \( P_{\text{MAXRS}}(Q_x) \geq P_t \), which is called "refinement". Unfortunately, in Theorem 1, we have shown that such verification procedure is \#P-hard. To tackle the hardness, we propose a sampling based algorithm. We first introduce the basic idea of this sampling-based algorithm called VERIFY in Section 4.1. Then, in Section 4.2, we analyze the error bound of our proposed sampling based approach. Specifically, by setting the sample size appropriately, the error of our sampling approach can be bounded with high success probability. Note that, the reasons that our sampling-based algorithm significantly accelerates the refinement step are threefold: 1) it avoids enumerating all possible worlds to verify whether \( P_{\text{MAXRS}}(Q_x) \geq P_t \); 2) the sample size could be controlled with the precision guarantee and 3) instead of considering all possible candidate regions, we only focus on the regions selected by Algorithm 2.

**Algorithm 3: VERIFY**

**Input:** a candidate region set \( C \), an uncertain database \( \mathcal{U} \), and a probability threshold \( P_t \)

**Output:** a PMaxRS solution set \( y' \)

/* initialization */
1. \( s \leftarrow O\left(\frac{1}{\log \left(2p_{22} \cdot \log(1-p_{22})\right)}\right) \)
2. \( \text{freq}([Q_x]) \leftarrow 0 \) for all \( Q_x \in C \);
3. \( y' \leftarrow \phi \)
4. /* begin sampling */
5. sample a possible world \( pw \) according to \( \mathcal{U} \)
6. /* get approximated max range-sum */
7. \( \sum' \leftarrow \max_{Q_x \in C}(Q_x, \sum \text{pw}) \)
8. for \( Q_x \in C \) do
   9. if \( \text{freq}(Q_x) > s \) then
      10. \( \text{freq}(Q_x) \leftarrow \text{freq}(Q_x) + 1 \)
      11. \( y' \leftarrow y' \cup \{Q_x\} \)
12. return \( y' \)

**4.1 Sampling Based Refinement**

The basic idea is that we sample \( s \) representative possible worlds, denoted as \( pw_1, pw_2, \cdots, \) and \( pw_s \), according to their occurrence probabilities. Then, to verify whether a given candidate region \( Q_x \) is in PMaxRS solution, instead of summarizing all possible-world probabilities as shown in Definition 5, we count the frequency that \( Q_x \) becomes the region whose range-sum is maximal on the given
possible world \( pw \). For each region, if its frequency, denoted by \( f\text{req}\{Q_x\} \), is larger than \( s \cdot P_t \), then the region \( Q_x \) is selected into the final \( \text{PMaxRS} \) solution set. That is to say, we use value \( \frac{1}{\delta_1} f\text{req}\{Q_x\} \) as an approximation of the true \( \text{PMaxRS} \) probability \( Pr_{\text{maxrs}}(Q_x) \).

Our sampling based refinement algorithm, called \( \text{Verify} \), is shown in Algorithm 3. The input parameters of \( \text{Verify} \) are: 1) the set of candidate regions \( C \) returned by \( \text{CandidateGen} \) shown in Algorithm 2; 2) a probabilistic spatial database \( U \), and 3) a probabilistic threshold \( P_t \). Lines 1-3 initialize the sample size \( s \), a PMaxRS solution set \( Y^* \) and a frequency list \( f\text{req}\{Q_x\} \) which keeps the count that \( Q_x \) holds the maximum range-sum through \( s \) samplings. Specifically, the sample size \( s \) is set to \( O \left( \max \left( \frac{4}{\epsilon^2 \log \frac{2}{\delta_1}, \frac{\log(n - 1 - \delta_1)}{\log(n - 1 - N^{K - P_t})} \right) \right) \)

where \( \epsilon, \delta_1, \delta_2 \) are three accuracy parameters and \( K \) is the average number of instances per uncertain object. We will show later that such a setting of \( s \) guarantees the sample accuracy of Algorithm 3. From line 4, we begin the sampling procedure. In each iteration, we sample a possible world and update the frequency list \( f\text{req}\{1 \cdots |C|\} \). Once \( f\text{req}\{Q_x\} \) exceeds \( s \cdot P_t \), we add \( Q_x \) into the solution set \( Y^* \). The iteration above will be repeated \( s \) times and after it terminates, we return \( Y^* \) as the PMaxRS solution. Note that, theoretically, \( \sum^* \) in line 6 should be the maximum range-sum over sampled possible world \( pw \). However, considering algorithm efficiency, we use \( \max_{Q_x \in C} Q_x, \text{sum}(pw) \) instead of the actual value of \( \sum^* \) as an approximation. That is to say, we ignore all the regions filtered out by Algorithm 2.

**Complexity Analysis.** To efficiently sample \( s \) possible worlds, for each uncertain object \( U_j \in U \), we sample its \( s \) instances \( u_i^{(1)} \cdots u_i^{(s)} \) based on their probabilities with replacement. Then, \( pw_j = \{u_i^{(j)} | i = 1 \cdots s \} \) and the probability that \( pw_j \) is sampled is \( \prod_{i=1}^{s} p_i^{(j)} \). Such a sampling procedure can be done in \( O(sn \log s) \) by using appropriate sorting-based sampling technique. The calculation of range-sum and the update of \( f\text{req}\{Q_x\} \) for all \( Q_x \in C \) can be done within time \( O(s|C|) \), where \( |C| \) is the cardinality of the candidate set \( C \). Thus, the total time complexity is \( O(s|C|) \), where \( s \) is the sample size. Note that, in the worst case, \( |C| \) can be \( O(N^2 K^2) \). However, the experimental results show that by using our pruning technique, most of (more than 95%) possible regions can be safely filtered out. Thus, the quadratic cardinality of \( |C| \) is not a big issue to the performance of Algorithm 3.

### 4.2 Sampling Error Analysis

In this subsection, we analyze the error introduced by random sample. The major theoretical result is: by setting the sample size \( s \) is \( O \left( \max \left( \frac{4}{\epsilon^2 \log \frac{2}{\delta_1}, \frac{\log(n - 1 - \delta_1)}{\log(n - 1 - N^{K - P_t})} \right) \right) \) as shown in Algorithm 3, the probability that the verification of \( Q_x \) fails can be bounded within \( \min(\delta_1, \delta_2) \), where \( \delta_1 \) and \( \delta_2 \) are two accuracy parameters specified by user. To prove that, we observe two facts in Algorithm 2 that introduce errors:

(i) line 10 uses \( \frac{1}{\delta_1} f\text{req}\{Q_x\} \) to estimate \( Pr_{\text{maxrs}}(Q_x) \);

(ii) line 6 uses \( \max_{Q_x \in C} Q_x, \text{sum}(pw) \) as an approximation of the maximum range-sum over sampled possible worlds.

We first analyze the error caused by Case (i). By selecting an appropriate sample size \( s_1 \), the absolute error between \( Pr_{\text{maxrs}}(Q_x) \) and its estimation \( \frac{1}{\delta_1} f\text{req}\{Q_x\} \) can be bounded. The error bound is shown in Theorem 2.

**Theorem 2.** Given a region \( Q_x \), by choosing the sample size \( s_1 = O \left( \frac{4}{\epsilon^2 \log \frac{2}{\delta_1}} \right) \), for all \( \epsilon, \delta_1 > 0 \), it always holds that,

\[
Pr \left[ \left| \frac{1}{\delta_1} f\text{req}\{Q_x\} - Pr_{\text{maxrs}}(Q_x) \right| \leq \epsilon \right] \geq 1 - \delta_1,
\]

where \( f\text{req}\{Q_x\} \) is the frequency that region \( Q_x \) has the highest range-sum over \( s_1 \) sampled possible worlds (i.e., the frequency that \( Q_x \) is the exact MaxRS solution over \( s_1 \) possible worlds).

Then, for Case (ii), we prove that, by selecting an appropriate sample size \( s_2 \), the ignorance of all the non-candidate regions when we calculate the maximum range-sum \( \sum^* \) introduces low errors that can be bounded. The details are shown in Theorem 3.

**Theorem 3.** Given a set of candidate regions \( C \), a set of non-candidate regions \( \bar{C} \), and a set of sampled possible worlds \( PW^* = \{pw_1, pw_2, \cdots, pw_{s_2}\} \), by setting the sample size \( s_2 = O \left( \frac{\log(1/\delta_2)}{\log(1 - N^{K - P_t})} \right) \), for all \( \delta_2 > 0 \), it always holds that,

\[
Pr \left\{ \bigwedge_{pw \in PW^*} \bigwedge_{Q_x \in C} Q_x, \text{sum}(pw) \neq s^*_pw \right\} \geq 1 - \delta_2,
\]

where \( s^*_pw \) is the maximum range-sum value over possible world \( pw \).

Thus, by combining Theorem 2 with Theorem 3, in Algorithm 3, we set the sample size to the maximum value between \( s_1 \) and \( s_2 \), i.e., \( s = O \left( \max \left( \frac{4}{\epsilon^2 \log \frac{2}{\delta_1}, \frac{\log(n - 1 - \delta_1)}{\log(n - 1 - N^{K - P_t})} \right) \right) \), which guarantees that the success probability of Algorithm 3 is at least \( 1 - \min(\delta_1, \delta_2) \). Notice that, the first term of \( s \), i.e., \( \frac{4}{\epsilon^2 \log \frac{2}{\delta_1}} \), is independent of the parameters of a \( \text{PMaxRS} \) query and only depends on \( \epsilon \) and \( \delta_1 \), which controls the estimation error of the \( \text{PMaxRS} \) probability. As for the second term, \( \frac{\log(1/\delta_2)}{\log(1 - N^{K - P_t})} \) dominates the sample size \( s \) only when \( NK \) or \( P_t \) takes very small value. This is reasonable since there are two cases leading to high error by neglecting of all non-candidate regions: (1) the size of the given database is very small (corresponding to small \( NK \)) and (2) the tolerance of sample errors is extremely low (corresponding to small \( P_t \)).

**Discussions of the extensibility.** So far, we have discussed the details of our \( P\text{MaxRS}_\text{Framework} \). By slight modification, the \( P\text{MaxRS}_\text{Framework} \) can also be used to answer variants of the \( P\text{MaxRS} \) query, such as \( \text{PMaxCRS} \) (a circular version of \( \text{PMaxRS} \)) and top-\( k \) \( \text{PMaxRS} \) queries. For more details about the Extensibility of our framework, please refer to Appendix H.

## 5 EXPERIMENTAL STUDY

In this section, we conduct extensive experiments to demonstrate the effectiveness and efficiency of \( P\text{MaxRS}_\text{Framework} \) on several generated datasets with various practical parameter settings. All the experiments were conducted on a Linux server with Intel(R) Xeon(R) CPU X5675 @ 3.07GHz and 32GB memory, and all the algorithms were implemented in Java.

### 5.1 Competitors

To demonstrate that our \( P\text{MaxRS}_\text{Framework} \) is both effective and efficient, we select two baseline algorithms for comparisons. The first baseline is Expected MaxRS, which is denoted as “EMaxRS.”
EMaxRS returns the region \( Q_x^* \) such that its expected range-sum, \( E[Q_x, \text{sum}] \), is maximized. To implement EMaxRS algorithm, we can apply the precise MaxRS algorithm to a modified certain database consisting of all instances associated with weights \( p \times w \), that is:

\[
Q_x^* = \arg \max_{x \in \mathbb{R}^d} \sum_{u_i \in U_x} \sum_{l_j \in Q_x} p_{ij} \times w_{ij}.
\]  

(9)

By selecting appropriate precise MaxRS algorithms like [13, 15, 24], EMaxRS has the time complexity \( O(NK \cdot \log(NK)) \), where \( N \) is the total number of uncertain objects and \( K \) is the average number of instances per uncertain object. Our implementation of precise MaxRS algorithm is based on an S-tree [15], which is a sweep-line algorithm and has been proved to be optimal in main memory.

The second competitor is called APPROX, which is a sampling based approximation algorithm of the PMaxRS problem. APPROX is shown in Algorithm 4. In line 1, the sample size \( s \) is set to \( \frac{1}{\epsilon^4} \log\frac{1}{\delta} \). Line 2 initializes an empty hashmap which stores the key-value pair as \( (\text{region}, \text{count}) \), where \( \text{region} \) denotes the possible region in PMaxRS solution set and \( \text{count} \) denotes how many times this region has maximum range-sum among all sampling results. Lines 3-7 repeat the sampling procedure \( s \) times. In each iteration, we sample a possible world \( pw \) in the same way used in Algorithm 3. Then, we invoke the precise MaxRS algorithm to get the regions with the highest range-sum and update the count dictionary. Finally, we return all the regions in the dictionary with \( \text{count} \) value larger than \( s \cdot P_f \). Note that, the sample error of APPROX is guaranteed by Theorem 2 and due to the limitation of space, we do not analyze it again. The total running time of Algorithm 4 contains two parts: (1) time for generating \( s \) possible worlds and (2) time for running the precise MaxRS algorithm \( s \) times over a database with size \( N \).

Algorithm 4: APPROX

Input: an uncertain database \( U \), a region size \( a \times b \) and a probability threshold \( P_f \).

Output: the PMaxRS solution set

1. \( s \leftarrow O(\frac{1}{\epsilon^4} \log\frac{1}{\delta}) \).
2. \( \text{cnt\_dict} \leftarrow \text{empty\_dictionary} \).
3. \( \text{// begin sampling} \)
4. for \( i = 1 \) to \( s \) do
5. sample a possible world \( pw \) according to \( U \);
6. run precise MaxRS on \( pw \) with size parameter \( a \times b \) and record the result as \( Q' \);
7. if \( Q' \) is key of \( \text{cnt\_dict} \) then
8. \( \text{cnt\_dict}[Q'] \leftarrow \text{cnt\_dict}[Q'] + 1 \);
9. else add \( (Q', 1) \) to \( \text{cnt\_dict} \);
10. return \( \{Q' | \text{cnt\_dict}(Q') \geq s \cdot P_f \} \).

5.2 Dataset Description

Real Spatial datasets. We use two real spatial datasets\(^3\): Long Beach’s country roads data (denoted by LB) and LA rivers and railways from Tiger/Line (denoted by RR), which contain 53,145 minimum bounding rectangles (MBRs) and 128,971 MBRs, respectively. Since each spatial object is represented as a MBR, to simulate the uncertainty, for each MBR, we uniformly generate 5 samples within the MBR as its instances. Then, each sample is assigned with a value from \( (0, 1) \) as its appearance probability, which will be normalized such that the probabilities of all samples of each uncertain object sum to 1. Note that, all the locations are rescaled to range \([0, 10000] \times [0, 10000] \).

Synthetic datasets. For synthetic datasets, we generate uncertain objects with various parameter values by following the convention in previous literature about probabilistic spatial database such as [21]. Uniform and Gaussian distributions are adopted to generate \( N \) spatial points on \( \mathbb{R}^2 \) space (also within range \([0, 10000] \times [0, 10000] \)), which are regarded as “center” points of uncertainty regions. After obtaining \( N \) spatial center points, we draw \( K \) possible instances for each object. Different distributions are used to make the experimental results more convincing. We use a length-range parameter \( r \) to control the shape of data distribution. For uniform distribution, we generate \( K \) samples within rectangular region \([x - r/2, x + r/2] \times [y - r/2, y + r/2] \), where \((x, y)\) is the coordinate of center point. Similarly, for Gaussian distributions, we draw \( K \) instances by sampling from a 2-dimensional Gaussian distribution with mean \((x, y)\) and variance \(r^2/2\sqrt{3}\). We use \( UU \) (or \( UG \)) to denote the dataset with Uniform distributed centers and Uniform (or Gaussian) distributed instances. Similarly, \( GU \) (or \( GG \)) means the dataset with Gaussian distributed centers and Uniform (or Gaussian) distributed instances.

Parameter Settings. The parameter settings are shown in Table 5. Each time, we vary one parameter, while other parameters are set to the underlined default values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>#uncertain objects</td>
<td>10K, 20K, 50K, 100K, 500K</td>
</tr>
<tr>
<td>( K )</td>
<td>#instances per object</td>
<td>2, 3, 5, 8, 10</td>
</tr>
<tr>
<td>( r )</td>
<td>length-range of object</td>
<td>5, 10, 20, 30, 60</td>
</tr>
<tr>
<td>( s )</td>
<td>sample size</td>
<td>500</td>
</tr>
<tr>
<td>( P_f )</td>
<td>probabilistic objects</td>
<td>50</td>
</tr>
<tr>
<td>( a \times b )</td>
<td>size of query region</td>
<td>100 \times 100</td>
</tr>
</tbody>
</table>

5.3 Pruning Power

We first report the pruning power of our candidate generation algorithm shown in Algorithm 2. As we discussed in Section 3.2, different selection strategy of the reference region \( Q_A \) yields different pruning power. The reason is that, our pruning bound derived in Lemma 2 filters out the regions “worse” than \( Q_A \). Thus, an initially good \( Q_A \) with high PMaxRS probability will enhance the pruning power. Three heuristic strategies have been discussed at the end of Section 3.2: (1) MaxPossible: select \( Q_A \) as the MaxRS answer over the possible worlds with the highest probability; (2) \( k - \) Sampling: generate \( k \) (here, we set \( k \) to 10) possible worlds and select \( Q_A \) as the region with the highest frequency to be MaxRS solution over \( k \) possible worlds; and (3) EMaxRS: let \( Q_A \) be the answer of expected MaxRS.

To compare the pruning power, we list the number of all possible candidates and the number of candidates after pruning w.r.t. three different reference region selection strategies MaxPossible, \( k - \) Sampling and EMaxRS. We test the pruning power over four synthetic datasets \( GG, GU, UG \) and \( UU \) where \( N \) is 10K and other parameters are set to their default values. Table 6 shows the experimental results of pruning power. The results are shown in Table 6.
We can see that the pruning ratios on four datasets are all high.

For \( k - \) Sampling and EMaxRS strategies, more than 99% candidates are pruned; and for MaxPossible, more than 93% candidates are filtered out on all test datasets. Moreover, the pruning performance is robust for different data distributions, including both center point distribution and instance distribution. Note that, due to the locality of the data distribution, which is very common in real spatial data, the pruning power of our candidate generation algorithm can be much better than that reported in Table 6.

From the results, using EMaxRS to initialize \( Q_A \) always obtains the best pruning power among all the three \( Q_A \) selection strategies (best on datasets GG, UG, and UU). The drawback of EMaxRS is that it costs more time than the other two approaches (can be shown in Section 5.6). As an alternative, the \( k - \) Sampling strategy runs much faster, but can still achieve high pruning power. Therefore, in the subsequent experiments, we will use the \( k - \) Sampling strategy to select the reference region \( Q_A \).

### 5.4 Sampling Accuracy

Our PMaxRS Framework is based on sampling and its accuracy is bounded in Theorems 2 and 3. We conduct experiments to test the sampling error. The absolute error between the actual PMaxRS probability and our sampling-based estimator is used as measurement. Since computing actual PMaxRS probability is intractable (\( \#P \)-hard), we run sampling with a large size \( s = 5,000 \) and regard the result as optimal. Then, we uniformly sample 50 points \( x_1, x_2, \ldots, x_{50} \) from \([0,10000] \times [0,10000] \). The absolute error of \( x_i \) is defined as \( \text{abs \ err}_i = |\text{freq}/s - \text{Pr}_{\maxrs}(Q_{x_i})| \), where \( \text{freq}/s \) is the estimated PMaxRS probability \( \text{Pr}_{\maxrs}(Q_{x_i}) \). Then, the total error, \( \text{abs \ err} = \sum_{i=1}^{50} \text{abs \ err}_i \), is used to measure the sampling accuracy. Figure 7 reports the convergence of abs \ err w.r.t. sample size \( s \) over four datasets GG, GU, UG, and UU with default parameter settings. From Figure 7, we can see that abs \ err converges swiftly and stays stable after the sample size \( s \) increases to 50 regardless of the data distribution, which can demonstrate that our sampling-based framework is correct and efficient.

Note that, the experimental results on several datasets show that the convergence rate is even much faster than that suggested by Theorem 2 and Theorem 3. For example, if we select the sampling accuracy parameters \( \epsilon = 0.1 \) and \( \delta_1 = \delta_2 = 0.1 \), the sample size is estimated as 500 by using formula \( s = O \left( \max \left( \frac{1}{\epsilon} \log \frac{2}{\delta_1} \log (1/\delta_2) \right) \right) \), which guarantees \( \text{abs \ err} < 5 \) with probability 0.9. However, from
### 5.6 The Efficiency of PMaxRS\_Framework

In this subsection, we report the efficiency and scalability of our PMaxRS\_Framework w.r.t. three parameters \( N, K \), and \( r \). We report the total time (denoted by Total), which consists of filtering time (denoted by Prune) and refinement time (denoted by Refinement), of PMaxRS, and the running time of competitors Approx and EMaxRS. All the experiments are conducted over four synthetic datasets GG, GU, UG, and UU by varying different parameters as shown in Table 5.

**Effect of \( N \)**. In Figures 6(a)–6(d), we first test the scalability of PMaxRS\_Framework over 4 synthetic uncertain datasets with respect to data size \( N = 10K, 20K, 50K, 100K, \) and 500K. On all datasets, the algorithm Approx is always the most costly one, and the performance of EMaxRS and our PMaxRS\_Framework is similar. Note that, the refinement time (Refinement) is much lower than the filtering time (Prune), which means the candidate generation process dominates the total time of PMaxRS\_Framework.

**Effect of \( K \)**. Figures 6(e)–6(h) illustrate the query performance of PMaxRS and baselines by varying the number of instances per uncertain object \( K = 2, 3, 5, 8, \) and 10. The running time of Approx is stable when \( K \) increases. This is because the time complexity of Approx only depends on the sample size \( s \) and the total number of uncertain objects \( N \). When \( K \) increases, both filtering time (Prune) and refinement time (Refinement) increase. That is because the total number of possible candidate regions increases as \( K \) increases. Besides, the calculation of statistics used in our pruning algorithm, \( \mu \) and MS, will be more costly when \( K \) increases.

**Effect of \( r \)**. In Figures 6(i)–6(l), we study the effect of the length-range parameter \( r \) by setting \( r = 5, 10, 20, 30, \) and 60. On four datasets, when \( r \) increases, the running times of Approx and EMaxRS are stable, since only \( N \) and \( K \) influence them. However, the filtering time (Prune) increases when \( r \) increases, which leads to the increase of the total running time of PMaxRS\_Framework. The reason is that, when \( r \) increases, the variance of possible instances’ distribution increases, which leads to instances spreading much wider and increases the possible appearance of the correlated case discussed in Section 3.3.2. According to our solution to the

---

Table 7: PMaxRS\_Framework results on different datasets.

<table>
<thead>
<tr>
<th></th>
<th>top-1</th>
<th>top-2</th>
<th>top-3</th>
<th>top-4</th>
<th>top-5</th>
<th>EMaxRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB</td>
<td>(2569.5, 1418.7)</td>
<td>(2573.1, 1398.9)</td>
<td>(582.0, 1401.5)</td>
<td>(5268.0, 1401.5)</td>
<td>(5268.0, 1401.5)</td>
<td>(5268.0, 1401.5)</td>
</tr>
<tr>
<td>RR</td>
<td>(200.5, 1940.8)</td>
<td>(198.1, 1940.8)</td>
<td>(198.1, 1940.8)</td>
<td>(200.5, 1940.8)</td>
<td>(198.1, 1940.8)</td>
<td>(200.5, 1940.8)</td>
</tr>
<tr>
<td>GG</td>
<td>(2529.7, 5392.2)</td>
<td>(2530.2, 5386.8)</td>
<td>(2530.2, 5386.8)</td>
<td>(2530.2, 5386.8)</td>
<td>(2530.2, 5386.8)</td>
<td>(2530.2, 5386.8)</td>
</tr>
<tr>
<td>GU</td>
<td>(5303.3, 5382.8)</td>
<td>(5303.3, 5382.8)</td>
<td>(5303.3, 5382.8)</td>
<td>(5303.3, 5382.8)</td>
<td>(5303.3, 5382.8)</td>
<td>(5303.3, 5382.8)</td>
</tr>
<tr>
<td>UG</td>
<td>(2590.0, 5759.5)</td>
<td>(2590.0, 5759.5)</td>
<td>(2590.0, 5759.5)</td>
<td>(2590.0, 5759.5)</td>
<td>(2590.0, 5759.5)</td>
<td>(2590.0, 5759.5)</td>
</tr>
<tr>
<td>UU</td>
<td>(2490.1, 5429.6)</td>
<td>(2490.1, 5429.6)</td>
<td>(2490.1, 5429.6)</td>
<td>(2490.1, 5429.6)</td>
<td>(2490.1, 5429.6)</td>
<td>(2490.1, 5429.6)</td>
</tr>
</tbody>
</table>

---

The experimental results, abs_error is less than 1 for sample size \( s \geq 50 \), which is much smaller than 500.

Figure 7: Illustration of the sample error convergence.

---

### 5.5 PMaxRS\_Framework Results

In this subsection, to show that our PMaxRS\_Framework is effective, we report results with top-5 highest PMaxRS probabilities and compare them with the EMaxRS result. Table 7 presents the results returned by PMaxRS\_Framework and EMaxRS over 5 datasets RR, LB, GU, GU, GG with default parameter settings. We use the coordinates of the center point of a rectangle to represent the corresponding region and use RS to denote the range-sum of the result region over certain dataset.

In Table 7, each row denotes results on one dataset. The first five columns present the results from our PMaxRS\_Framework. We select to report results with top-5 PMaxRS probabilities since according to our observation, by setting the probability threshold \( P_L \) as 0.01, top-5 regions usually have significantly high confidences (PMaxRS probabilities). Then, the 6th column in Table 7 show the results returned by EMaxRS. Note that, for dataset RR, only two results are returned by our PMaxRS\_Framework, since there is high locality residing in dataset RR, which leads to only two regions having PMaxRS probability exceeding the threshold \( P_L \). Compared with EMaxRS results, our PMaxRS\_Framework is much better, since PMaxRS\_Framework returns several results with both high range-sum values and high confidences. In other words, different from EMaxRS which retrieves only one region with the highest expected range-sum, the outputs of PMaxRS\_Framework are much more diverse and contain many “sub-optimal” but insightful results. In real-world applications, these sub-optimal PMaxRS answers are very important for decision making. For example, for urban transportation managers, they care about not only the most likely jam region, but also other regions with relatively high confidences that traffic jam happens; that is, PMaxRS\_Framework provides multiple reasonable choices for decision makers.
correlated case, we need to transform the reference region \( Q_A \), and the statistical information of the transformed region should be re-evaluated, which leads to the increase of the filtering time (Prune).

In summary, our \( \text{PMaxRS\_Framework} \) is effective and efficient in answering PMaxRS queries over uncertain database. Note that, due to the space limitation, other parameter settings like more types of data distributions have experimental results similar to Figure 6, and thus are omitted here.

6 RELATED WORKS

In this section, we will review related works on the MaxRS problem and uncertain data management, and compare them with our work.

**The MaxRS Problem.** As an important tool to find Regions of Interest (ROIs), the MaxRS query aims to retrieve regions with the highest range-sum. Due to its usefulness in location-based services, the MaxRS problem has been well investigated in [13, 29]. However, MaxRS was first discussed by the computational geometry community [18, 24] and they proved that MaxRS is equivalent to the Rectangle Stabbing problem which transforms spatial objects to rectangles centering at corresponding positions and then retrieves the region where the most rectangles intersect. With such transformation, the basic idea to solve MaxRS is to maintain a sweep-line and record the maximum overlapping area the line has swept. By using appropriate index structure like \( aSB-tree \) [15], the sweep-line algorithm terminates in \( O(n \log n) \), which is optimal among all of the comparison-based algorithms. To reduce the I/O consumption, [13] proposed a scalable MaxRS algorithm and to accelerate MaxRS query processing with a little loss of precision, Tao et al. proposed a grid sampling based approximation algorithm which reduces the time complexity to \( O(n \log^2 n + n \log \log n) \) [29]. Note that, our candidate generation algorithm also follows the idea of transformation to the rectangle intersection problem. However, the intrinsic difference is that, we do the transformation for traversing the whole data space to find any possible candidate region, which means the previous index structure like \( aSB-tree \) cannot be used in our \( \text{PMaxRS\_Framework} \) since it only records the region with highest range-sum but ignore all the other regions. Besides the original MaxRS problem, recently, researchers also focus on variants of the MaxRS problem such as the rotating MaxRS [10] where the query regions can rotate with an angle and the dynamic MaxRS [5, 6] where the spatial objects might move and a monitoring algorithm is put forward to update the current MaxRS result.

**Uncertain Data Model.** In the literature [3, 7, 8, 11, 12, 25, 27, 28, 32], uncertain data management and query processing are important due to the universal existence of uncertainty in real world. Many techniques have been designed for extending traditional database techniques to uncertain case. Specifically, the related topics include uncertainty modeling [3], uncertain data indexing [7, 12, 28], uncertain data mining [8, 32], and query processing over uncertain database such as uncertain top-\( k \) [27] and probabilistic kNN queries [11].

Note that, there is a paper also dealing with the probabilistic MaxRS query [23]. However, there are two main problems in [23], First, it can only deal with the weight uncertainty and fail to consider the location uncertainty, whereas our \( \text{PMaxRS\_Framework} \) provides a much more generic uncertain data model. Second, this work returns all the regions with PMaxRS probability larger than 0, which may return numerous results that are not useful due to low confidences. Therefore, with different data models and problem settings, we cannot directly apply previous techniques to tackle our PMaxRS problem.

7 CONCLUSION

In this paper, we propose a novel query, called the Probabilistic MaxRS query, which retrieves \( a \times b \) rectangular regions \( Q_x \) such that the probability that \( Q_x \) has the maximum range-sum exceeds a probability threshold \( P_t \). We illustrate that the PMaxRS query is useful for retrieving regions of interest and location-based services. Since answering such a query is challenging and direct evaluation is intractable, we develop an efficient \( \text{PMaxRS\_Framework} \) which is based on pruning and refinement strategies. Finally, by using real and synthetic datasets, we demonstrated the effectiveness and efficiency of our proposed algorithms.

REFERENCES

A PROOF OF LEMMA 1

Proof. According to Eq. (2), we have:

\[ Pr_{\text{max}}(x) = \sum_{p \in PW} Pr(p_{\omega}) \cdot \mathbb{I}(Q_{x, \text{sum}}(p_{\omega}) \geq Q_{x', \text{sum}}(p_{\omega})) \]

where \( \mathbb{I}(z) \) is a logical indicator function: if \( z = \text{True} \), then \( \mathbb{I}(z) = 1 \); otherwise, \( \mathbb{I}(z) = 0 \). Thus, we complete the proof. □

B PROOF OF LEMMA 2

Proof. According to Lemma 1, we have:

\[ Pr_{\text{max}}(Q_x) = Pr\left( \bigwedge_{x'} Q_{x, \text{sum}} \geq Q_{x', \text{sum}} \right) \]

\[ \leq Pr\left( Q_{x, \text{sum}} \geq Q_{A, \text{sum}} \right) \]

\[ = Pr\left( Q_{x, \text{sum}} - Q_{A, \text{sum}} - \mu \geq -\mu \right) \]

Then, we use Cantelli’s Inequality [14], that is, for a random variable \( X \) and a positive number \( \lambda \), it holds that:

\[ Pr[X - \mu_X \geq \lambda] \leq \frac{\sigma_X^2}{\lambda^2 + \sigma_X^2} \]

(12)

where \( \mu_X \) and \( \sigma_X^2 \) are expectation and variance of \( X \) respectively. Combining Eq. (11) with Eq. (12), set \( X = Q_{x, \text{sum}} - Q_{A, \text{sum}} \) and \( \lambda = -\mu \). Since \( -\mu = -E[Q_{x, \text{sum}} - Q_{A, \text{sum}}] = E[Q_{A, \text{sum}} - E[Q_{x, \text{sum}}]] \), if \( E[Q_{x, \text{sum}}] < E[Q_{A, \text{sum}}] \), then \( \lambda = -\mu > 0 \). Thus, the condition of Cantelli’s Inequality is satisfied and we have:

\[ Pr_{\text{max}}(Q_x) \leq Pr\left( Q_{x, \text{sum}} - Q_{A, \text{sum}} - \mu \geq -\mu \right) \]

\[ \leq \frac{\sigma^2}{\sigma^2 + \mu^2} \]

(13)

Note that, the condition of Cantelli’s Inequality cannot be always satisfied. If \( E[Q_{x, \text{sum}}] \geq E[Q_{A, \text{sum}}] \), \( \lambda = -\mu \leq 0 \), which breaks the condition of Cantelli’s Inequality, then the upper bound of \( Pr_{\text{max}}(Q_x) \) is set to 1, which is the maximum possible value of a probability. This way, we prove the upper bound of \( Pr_{\text{max}}(Q_x) \) in 2 cases of Eq. (5). □

C PROOF OF LEMMA 3

Proof. Let the upper bound shown in Lemma 2 be larger than the probabilistic threshold \( P_t \). If \( \mu_{Q_x} \geq \mu_{Q_A} \), the upper bound of \( Pr_{\text{max}}(Q_x) \) equals to 1, which is larger than \( P_t \). Thus, \( Q_x \) is selected as a candidate, and we get the Case (i). For the Case (ii), if \( \mu_{Q_x} < \mu_{Q_A} \), the upper bound of \( Pr_{\text{max}}(Q_x) \) is \( P_t \cdot (\mu_{Q_x} - \mu_{Q_A})^2 \). Simplify it, we have:

\[ \frac{1}{1 - P_t} P_t^2 \sigma_{Q_x}^2 - 2P_t \cdot \mu_{Q_A} P_t \sigma_{Q_x}^2 + P_t - P_t^2 \sigma_{Q_x}^2 - MS - \sigma_{Q_A}^2 < 0. \]  

(15)

The LHS of Eq. (15) is a quadratic function of variable \( \mu_{Q_x} \) and the discriminant \( \Delta \) of its corresponding quadratic equation is:

\[ \Delta = \left( \frac{2P_t \cdot \mu_{Q_A}}{1 - P_t} \right)^2 - 4 \frac{P_t}{1 - P_t} \left( \frac{P_t}{1 - P_t} \sigma_{Q_A}^2 - MS - \sigma_{Q_A}^2 \right) \]

\[ = \frac{4}{1 - P_t} (MS + \sigma_{Q_x}^2 - P_t \cdot \mu_{Q_x}^2). \]

To satisfy the inequality shown in Eq. (15) with the constraint \( \mu_{Q_x} \in [0, \mu_{Q_A}] \), there should be \( \Delta > 0 \) and \( \mu_{Q_x} \in [\max(0, \mu_0), \min(\mu_{Q_A}, \mu_0)] \), where \( \mu_0^+ \) and \( \mu_0^- \) are the two zero points of the above quadratic function. After solving it, we get the Case (ii). □

D PROOF OF COROLLARY 1

Proof. Since \( Q_{A'} \) covers more uncertain objects than \( Q_{A} \), by following the similar procedures of proving Lemma 2, it holds that:

\[ Pr_{\text{max}}(Q_x) \leq Pr\left( Q_{x, \text{sum}} \geq Q_{A', \text{sum}} \right) \]

\[ \leq Pr\left( Q_{x, \text{sum}} \geq Q_{A', \text{sum}}^1 \right) \]

\[ \leq \frac{\sigma_{A', \text{sum}}^2}{\sigma_{A', \text{sum}}^2 + \mu_{A', \text{sum}}^2} \]

Thus, we complete the proof. □
E PROOF OF THEOREM 1

Proof. We give the sketch of the proof by using a reduction from the \#KNAPSACK problem [16]. According to Lemma 1, we can show that, for a given region \( Q_x \), the calculation of probability \( \Pr(Q_x \text{ sum} \geq Q'_{\text{sum}}) \), where \( Q'_{\text{sum}} \) is any region different from \( Q_x \), is a special case of calculating \( \Pr_{\text{max}}(Q_x) \). And similarly, \( \Pr(Q_x \text{ sum} \geq L) \), where \( L \) is an arbitrary positive constant, is also a special case of \( \Pr_{\text{max}}(Q_x) \). Then, according to [19, 26], the evaluation of \( \Pr(Q_x \text{ sum} \geq L) \) is \#P-hard by using a reduction from the \#KNAPSACK problem, which has been already shown to be \#P-complete [16]. Thus, the evaluation of \( \Pr_{\text{max}}(Q_x) \) must be \#P-hard. □

F PROOF OF THEOREM 2

Proof. For a given region \( Q_x \), we construct a random variable \( X \). If \( Q_x \) has the maximum range-sum over possible world \( pw_i \), where \( i = 1, 2, \cdots, |PW| \), then \( i^{th} \) possible value of \( X \) is 1, otherwise, is 0. Without loss of generality, let \( 1^{st} \sim k^{th} \) values be 1 and the others are 0. The distribution of \( X \) is shown in Table 8. Easily find that \( \Pr_{\text{max}}(Q_x) = E[X] \).

<table>
<thead>
<tr>
<th>( X )</th>
<th>( 0 )</th>
<th>( 1 )</th>
<th>( 0 )</th>
<th>( 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Pr )</td>
<td>( \Pr(pw_1) )</td>
<td>( \Pr(pw_2) )</td>
<td>( \Pr(pw_{k+1}) )</td>
<td>( \Pr(pw_{</td>
</tr>
</tbody>
</table>

Let \( X_1, X_2, \cdots, X_{s_1} \) be \( s_1 \) samples of a random variable \( X \). According to Chernoff-Hoeffding Theorem [17], we have:

\[
\Pr \left\{ \frac{1}{s_1} \sum_{i=1}^{s_1} X_i \geq E[X] + \epsilon \right\} \leq e^{-D(\epsilon / 2 || |E[X]| )},
\]

where \( D(\cdot || |E[X]| ) \) is K-L Divergence. Use bound: \( D((1 + x)p||p) \geq \frac{1}{2}x^2p \) for \( x \in [-\frac{1}{2}, \frac{1}{2}] \), we have:

\[
\Pr \left\{ \frac{1}{s_1} \sum_{i=1}^{s_1} X_i \geq E[X] + \epsilon \right\} \leq e^{-\frac{\epsilon^2}{4E[X]}} \leq e^{-\epsilon^2 / 4E[X]}. \tag{17}
\]

Similarly, we have,

\[
\Pr \left\{ \frac{1}{s_1} \sum_{i=1}^{s_1} X_i \leq E[X] - \epsilon \right\} \leq e^{-\epsilon^2 / 4E[X]}. \tag{18}
\]

Combining Eqs. (18) and (19), we have:

\[
\Pr \left\{ \frac{1}{s_1} \sum_{i=1}^{s_1} X_i \leq E[X] | E[X] \leq \epsilon \right\} \geq 1 - 2e^{-\epsilon^2 / 4E[X]}. \tag{20}
\]

By substituting \( s_1 \) with \( \frac{1}{\epsilon^2} \log \frac{2}{\delta_1} \), we finally have \( \Pr_{\text{max}}(Q_x) = \frac{1}{s_1} \sum_{i=1}^{s_1} X_i - E[X] | E[X] \leq \epsilon \geq 1 - \delta_1 \), which is equivalent to \( \Pr(| \frac{1}{s_1} \sum_{i=1}^{s_1} \text{freq}(Q_x) - \Pr_{\text{max}}(Q_x)| \leq \epsilon \geq 1 - \delta_1 \). □

G PROOF OF THEOREM 3

Proof. For any non-candidate region \( Q_x \) \( \not\in \mathcal{C} \), since \( Q_x \) is filtered out, there must be \( \Pr_{\text{max}}(Q_x) < P_t \). Combining Eq. (2), for a sampled possible world \( pw_i \in PW^s \), we have:

\[
\Pr \left\{ Q_x \text{ sum}(pw_i) = s_{pw_i} \right\} = \delta(Q_x | pw_i) \cdot \Pr(pw_i) \leq \sum_{pw \in PW} \delta(Q_x | pw) \cdot \Pr(pw) \leq \Pr_{\text{max}}(Q_x) < P_t.
\]

By using the union bound of probability, we have,

\[
\Pr \left\{ \bigwedge_{Q_x \in \mathcal{C}} Q_x \text{ sum}(pw_i) \neq s_{pw_i} \right\} = 1 - \Pr \left\{ \bigvee_{Q_x \in \mathcal{C}} Q_x \text{ sum}(pw_i) = s_{pw_i} \right\} \tag{22a}
\]

\[
\geq 1 - \sum_{Q_x \in \mathcal{C}} \Pr \left\{ Q_x \text{ sum}(pw_i) = s_{pw_i} \right\} \tag{22b}
\]

\[
\geq 1 - \frac{NK(NK - 1)}{2} P_t \geq 1 - (NK)^2 P_t. \tag{22c}
\]

Note that, from Eq. (22b) to Eq. (22c), we utilize the fact that there are at most \( \frac{N - K}{N - K - 1} \) possible candidate regions, where \( K \) is the average number of instances per uncertain object.

Then, since possible worlds \( pw_i \in PW^s \) are independently sampled, we have,

\[
\Pr \left\{ \bigwedge_{pw \in PW^s} \bigwedge_{Q \in \mathcal{C}} Q \text{ sum}(pw) \neq s_{pw} \right\} \geq (1 - (NK)^2 P_t)^{s_2}.
\]

Let \( s_2 = O \left( \frac{\log(1 - \delta_2)}{\log(1 - N^{-K} P_t)} \right) \), and we have the error bound shown in Eq. (8). Thus, we complete the proof. □

H EXTENSION OF PMAXRS FRAMEWORK

We briefly show how to extend our \( P\text{Mmaxs} \_\text{framework} \) to answer the Probabilistic Maximum Circular Range-Sum (PMaxCRS) queries and top-\( k \) PMaxRS queries.

PMaxCRS problem is a variant of the original PMaxRS query where the query region is a circle. PMaxCRS aims to retrieve a set of circles \( C_x \) with a user-specified radius such that the probability that \( C_x \) has the highest range-sum exceeds a given threshold. Note that, our pruning bound and sampling-based refinement algorithm not only hold for rectangular region, but also for circular region, or even arbitrary polygonal region, which means we can directly interchange rectangular region and circular region in Algorithms 2 and 3. The only difference is that, the traversing method should change to find all possible candidates (represented by overlapped circles). The traversing problem in circular case is related to the circle intersection problem and we can borrow the idea of overlap table in [31], which is originally used to support answering Bichromatic Reverse Nearest Neighbor (BRNN) queries.

A top-\( k \) PMaxRS query is also a variant of PMaxRS query which outputs \( k \) regions with top-\( k \) PMaxRS probabilities. The top-\( k \) PMaxRS query is useful, since instead of returning all the regions satisfying the threshold constraint, sometimes we only care about those with the highest PMaxRS probabilities. To support this operator, we can first set the probability threshold \( P_t \) to a relatively
small value (according to our tests, 0.01 is appropriate for most cases) to guarantee there are enough candidates returned by the candidate generation step. Then, we do the refinement similar to the original PMaxRS query. The difference is that, we return not only regions passing the refinement but also their corresponding PMaxRS probabilities which can be estimated via sampling during the refinement step. Then, we sort these regions by their PMaxRS probabilities and return the top-\(k\) regions.