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

Qiyu Liu, Xiang Lian and Lei Chen

June 12, 2019

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 γ^* 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 *#KNAPSACK* problem, which is the counting version of the classic *knapsack* problem and has already been proved as *#P-complete*. To solve that, we put forward an efficient *PMaxRS_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 $a \times b$ 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.

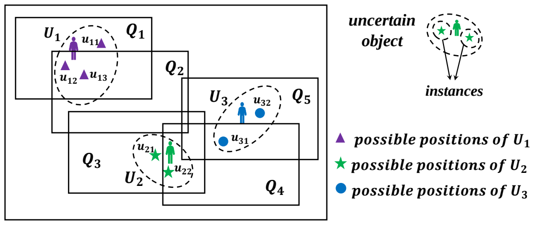


Figure 1: An example of the PMaxRS problem.

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 operator over an uncertain database, we propose a Probabilistic Maximum 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 *#KNAPSACK* problem, which has been shown to be *#P-complete* [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 1: Notations and descriptions.

Notation	Description
$\mathcal{U}$	an uncertain database
N	the number of uncertain objects
U_i	an uncertain object
$u_{i1}, \cdots, u_{iK}$	K instances of an uncertain object U_i
PW	the space of all possible worlds
pw	a possible world
γ^*	the solution set of the PMaxRS problem
P_t	the probability threshold
Q_x	a rectangular region centering at x
$Q_x.sum(pw)$	range-sum of Q_x over possible world pw
$\Pr_{maxrs}(Q_x)$	PMaxRS probability of given region Q_x

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 $O = \{o_1, o_2, \cdots o_N\}$, 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 Q_{x^*} centered at location x^* and with size $a \times b$ such that:*

$$Q_{x^*} = \underset{x \in \mathbb{R}^2}{\arg\max} \sum_{l_i \in Q_x} w_i, \tag{1}$$

where $l_i \in Q_x$ is the shorthand of l_i falls into the region Q_x.

Note that, we adopt the rectangular shape of region 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 $U_i = \{u_{i1}, u_{i2}, \cdots, u_{iK}\}$, where $u_{i1}, \cdots$, and 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},*[1] *with constraint $\sum_{j=1}^{K} p_{ij} = 1$.*

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

Definition 3. *(Possible World). A possible world $pw = \{u_{ij} | i = 1, 2, \cdots, N\}$ 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 $\Pr(pw) = \prod_{i=1}^{N} p_{ij}$.*[2] *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 $0.3 \times 0.5 \times 0.8 = 0.12$.

[1]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.
[2]By following the convention [21, 25, 27], we assume that uncertain objects in the database are independent of each other.

Table 2: An example of a probabilistic spatial database.

Uncertain Objects	Possible Instances	Weights	Probabilities
U_1	u_{11}	0.8	0.3
	u_{12}	0.5	0.6
	u_{13}	1.0	0.1
U_2	u_{21}	0.7	0.5
	u_{22}	0.4	0.5
U_3	u_{31}	0.9	0.8
	u_{32}	0.6	0.2

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 $\Pr_{maxrs}(Q_x)$, is defined as follows,*

$$\Pr_{maxrs}(Q_x) = \sum_{pw \in PW} \delta(Q_x||pw) \cdot \Pr(pw), \tag{2}$$

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

$$\delta(Q_x||pw) = \begin{cases} 1, & \text{if } Q_x.sum(pw) = \max_{\forall x'} Q_{x'}.sum(pw); \\ 0, & \text{otherwise,} \end{cases}$$

where $Q_x.sum(pw)$ denotes the range-sum of region Q_x on possible world pw, i.e.,

$$Q_x.sum(pw) = \sum_{\forall u_{ij} \in pw, l_{ij} \in Q_x} w_{ij}. \tag{3}$$

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, $\Pr_{maxrs}(Q_x)$ 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_i (i.e., $\Pr_{maxrs}(Q_i)$). As an example, Q_4 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 γ^* with size $a \times b$ such that for $\forall Q_x \in \gamma^*$, $\Pr_{maxrs}(Q_x) \geq P_t$; namely, the result of a PMaxRS query is:*

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

Table 3: An example of PMaxRS probabilities.

Region	Optimal on Possible Worlds	PMaxRS Probability
Q_1	$\{u_{11}, u_{21}, u_{32}\}\{u_{11}, u_{22}, u_{32}\}\{u_{13}, u_{22}, u_{32}\}$	0.07
Q_2	$\{u_{12}, u_{21}, u_{31}\}\{u_{12}, u_{21}, u_{32}\}\{u_{13}, u_{21}, u_{31}\}$ $\{u_{13}, u_{21}, u_{32}\}\{u_{13}, u_{22}, u_{32}\}$	0.36*
Q_3	None	0
Q_4	$\{u_{11}, u_{21}, u_{31}\}\{u_{11}, u_{22}, u_{31}\}\{u_{12}, u_{22}, u_{31}\}$ $\{u_{13}, u_{22}, u_{31}\}$	0.52**
Q_5	$\{u_{11}, u_{21}, u_{31}\}\{u_{12}, u_{22}, u_{32}\}$	0.18

*: PMaxRS probability≥ 0.3 **: PMaxRS probability≥ 0.5

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, $\Pr_{maxrs}(Q_x)$, 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 dicision 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 $\Pr_{maxrs}(Q_x)$ 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 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 $\Pr_{maxrs}(Q_x)$ 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 CandidateGen, denoted by C, in the refinement step shown in line 2, for each candidate region $Q \in C$, a subroutine Verify, shown in Section 4, checks whether it is indeed in PMaxRS solution set.

Algorithm 1: *PMaxRS_Framework*

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

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 $\Pr_{maxrs}(Q_x)$ 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 $\Pr_{maxrs}(Q_x)$ 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_{31}. 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 $p_{22} \times p_{31} = 0.4$. 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 $p_{22} \times p_{32} = 0.1$.

Table 4: Possible range-sums of Q_4 in Figure 1.

Q_4.sum	Probability
$w_{22} + w_{31} = 1.3$	$p_{22} \times p_{31} = 0.4$
$w_{22} + 0 = 0.4$	$p_{22} \times p_{32} = 0.1$
$0 + w_{31} = 0.9$	$p_{21} \times p_{31} = 0.4$
$0 + 0 = 0$	$p_{21} \times p_{32} = 0.1$

We then have the PMaxRS probability reduction lemma:

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

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

3.2 Pruning Bound

To derive the upper bound of $\Pr_{maxrs}(Q_x)$, 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 $\Pr_{maxrs}(Q_x)$:

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

$$\Pr_{maxrs}(Q_x) \leq \begin{cases} \frac{\sigma^2}{\sigma^2+\mu^2} & , \text{if } \mathbf{E}[Q_x.sum] < \mathbf{E}[Q_{\mathcal{A}}.sum] \\ 1 & , \text{otherwise}. \end{cases} \quad (5)$$

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

Lemma 2 provides an upper bound of $\Pr_{maxrs}(Q_x)$. Recall that, the PMaxRS query retrieves Q_x with $\Pr_{maxrs}(Q_x)$ 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 $Pr_{maxrs}(Q_x)$, 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 $\Pr_{maxrs}(Q_x)$, denoted as $ub\_P(Q_x)$. If $ub\_P(Q_x)$ 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 μ_{Q_x}, $\mu_{Q_{\mathcal{A}}}$, $\sigma^2_{Q_x}$ and $\sigma^2_{Q_{\mathcal{A}}}$ to denote $\mathbf{E}[Q_x.sum]$, $\mathbf{E}[Q_{\mathcal{A}}.sum]$, $\mathbf{Var}[Q_x.sum]$ and $\mathbf{Var}[Q_{\mathcal{A}}.sum]$, 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 that is independent with $Q_{\mathcal{A}}$, if one of the following conditions is satisfied, Q_x is selected as a candidate region.*

(i) $\mu_{Q_x} \geq \mu_{Q_{\mathcal{A}}}$, ***or***
(ii) $\mathbf{MS} > P_t \cdot \mu^2_{Q_{\mathcal{A}}} - \sigma^2_{Q_{\mathcal{A}}}$ ***and***

$$\mu_{Q_x} \in \left[\max\left(0, \frac{2P_t \cdot \mu_{Q_{\mathcal{A}}} - (1-P_t)\sqrt{\Delta}}{2}\right), \min\left(\mu_{Q_{\mathcal{A}}}, \frac{2P_t \cdot \mu_{Q_{\mathcal{A}}} + (1-P_t)\sqrt{\Delta}}{2}\right)\right],$$

where $\mathbf{MS}$ *is the mean square of $Q_x.sum$, that is, $\sigma^2_{Q_x} + \mu^2_{Q_x}$, and $\Delta = \frac{4}{1-P_t}\left(\mathbf{MS} + \sigma^2_{Q_{\mathcal{A}}} - P_t \cdot \mu^2_{Q_{\mathcal{A}}}\right)$.*

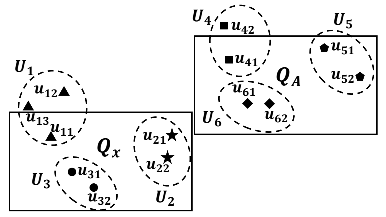


Figure 2: Example of case that Q_x and $Q_{\mathcal{A}}$ are independent.

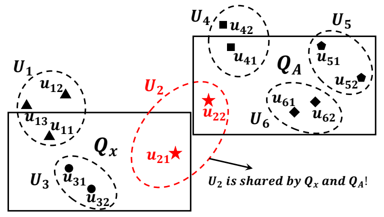


Figure 3: Example of case that Q_x and $Q_{\mathcal{A}}$ are correlated.

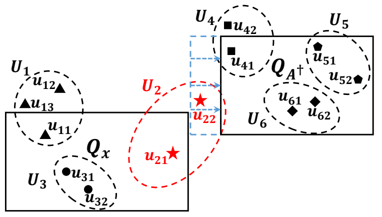


Figure 4: Transformation from correlated case to independent case.

3.3.2 Correlated Case. Now we discuss the correlated case where $Q_x.sum$ is correlated with $Q_{\mathcal{A}}.sum$. The reason leading to correlation is that Q_x and $Q_{\mathcal{A}}$ have mutually exclusive instances from the same uncertain objects. Figure 3 illustrates this case, where uncertain object U_2 intersects both Q_x and $Q_{\mathcal{A}}$. If U_2 takes instance u_{21}, it is impossible that u_{22} appears in $Q_{\mathcal{A}}$, and vice versa.

The correlation leads to significant increase of computation of σ^2, since we cannot decompose $\mathbf{Var}[Q_x.sum - Q_{\mathcal{A}}.sum]$ and need to enumerate all possible combinations of shared instances. When the correlation happens, i.e., Q_x and $Q_{\mathcal{A}}$ share common uncertain objects, one straightforward way to reduce the computation is to skip this region and directly set Q_x 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.

The idea is that, if an uncertain object is shared by Q_x and $Q_{\mathcal{A}}$, we ignore the instances of this object in $Q_{\mathcal{A}}$, and compute an upper bound of probability $\Pr_{maxrs}(Q_x)$ using rules for the independent case, which leads to a slightly loose upper bound. Figure 4 gives such an example. $Q_{\mathcal{A}}^\dagger$ is the transformed region which removes all shared uncertain objects. After the transformation, Q_x and $Q_{\mathcal{A}}^\dagger$ are independent. To guarantee the correctness of such approach, we have Corollary 1 showing that the upper bound of $\Pr_{maxrs}(Q_x)$ still holds by ignoring the correlated objects, which means the candidate selection rule under independent case shown in Lemma 3 can be still used on the transformed region.

Corollary 1. *(Correlated Candidate Selection). Suppose a correlated case where Q_x and $Q_{\mathcal{A}}$ share common uncertain objects and denote $Q_{\mathcal{A}}^\dagger$ as the transformed region of $Q_{\mathcal{A}}$, it holds that:*

$$\Pr_{maxrs}(Q_x) \le \begin{cases} 1 & , \text{ if } \mu^\dagger \ge 0; \\ \frac{\sigma^{\dagger 2}}{\sigma^{\dagger 2}+\mu^{\dagger 2}} & , \text{ otherwise}. \end{cases} \tag{6}$$

where $\mu^\dagger = \mathbf{E}[Q_x.sum - Q_{\mathcal{A}}^\dagger.sum]$ *and* $\sigma^{\dagger 2} = \mathbf{Var}[Q_x.sum - Q_{\mathcal{A}}^\dagger.sum]$.

Corollary 1 shows that the upper bound of the PMaxRS probability in the correlated case has the same formation as that in the independent case, which means we can use the candidate selection rule proposed in Section 3.3.1 by simply replacing $Q_{\mathcal{A}}$ with $Q_{\mathcal{A}}^\dagger$.

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 probabilistic 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 $O(N^2K^2)$, 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 $\frac{NK(NK-1)}{2} = O(N^2K^2)$ regardless of the distribution of data points. Hence, the lemma holds. □

Lemma 4 shows that there are at most $O(N^2K^2)$ possible candidate regions, which are represented as the overlapping areas in the transformed rectangle intersection problem. Thus, to find all candidate regions, we first transform the input probabilistic spatial database to a set of rectangles as discussed above. Then, a sweep-line from left to right is maintained to find out all the possible overlapping areas in the transformed rectangle intersection problem.

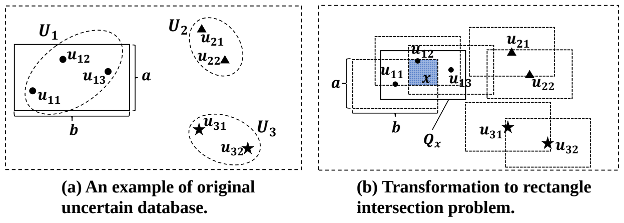


Figure 5: Transformation to rectangle intersection problem.

With all the details discussed above, we give the implementation of the candidate generation technique called CandidateGen 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 ℓ scans from left to right. At each state of ℓ, we invoke IsCandidate on the possible candidate regions generated by overlapping areas swept by ℓ and append the region which satisfies the candidate selection rules into C. The details of subroutine IsCandidate are shown in lines 8-12. Line 9 checks whether $Q_x.sum$ and $Q_{\mathcal{A}}.sum$ are correlated. If yes, we transform $Q_{\mathcal{A}}$ to $Q^{\dagger}_{\mathcal{A}}$ which shares no common uncertain objects with Q_x. Then, we invoke subroutine IndependentPruning, 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: CandidateGen

```
   Input: an uncertain database $\mathcal{U}$, a query size $a \times b$, a reference region $Q_{\mathcal{A}}$, and
          a probability threshold $P_t$
   Output: a set of candidate regions $C$
1  $\mathcal{R} \leftarrow$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 $Q$ generated by overlapping areas swept by $\ell$ do
5          if IsCandidate($Q$, $Q_{\mathcal{A}}$, $P_t$) then
6              $C \leftarrow C \cup \{Q\}$;
7  return $C$;
8  Function IsCandidate($Q_x$, $Q_{\mathcal{A}}$, $P_t$)
9      if $Q_x$ and $Q_{\mathcal{A}}$ share common uncertain objects then
10         $Q^{\dagger}_{\mathcal{A}}$ $\leftarrow$transformed region of $Q_{\mathcal{A}}$;
11         return IndependentPruning($Q_x$, $Q^{\dagger}_{\mathcal{A}}$, $P_t$);
12     else return IndependentPruning($Q_x$, $Q_{\mathcal{A}}$, $P_t$) ;
13 Function IndependentPruning($Q_x$, $Q_{\mathcal{A}}$, $P_t$)
14     calculate MS, $\mu_{Q_x}$, $\sigma^2_{\mathcal{A}}$ and $\mu_{\mathcal{A}}$;
15     if $\mu_{Q_x} > \mu_{Q_{\mathcal{A}}}$ then
16         return True;
17     if MS $> P_t \cdot \mu^2_{Q_{\mathcal{A}}} - \sigma^2_{Q_{\mathcal{A}}}$ and $\mu_V \in$
        $\left[\max(0, \frac{2P_t \cdot \mu_{Q_{\mathcal{A}}} - (1-P_t)\sqrt{\Delta}}{2}), \min(\mu_{Q_{\mathcal{A}}}, \frac{2P_t \cdot \mu_{Q_{\mathcal{A}}} + (1-P_t)\sqrt{\Delta}}{2})\right]$ then
18         return True;
19     return False;
```

Note that, there are several tricks that can accelerate CandidateGen. 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 IsCandidate is frequently invoked, for efficient calculation of **MS** and μ_{Q_x}, 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_4 shown in Figure 1 as an example, we can write down the *generating function* of $Q_4.sum$ as $\mathcal{F}_{Q_4} = (p_{22} \cdot x^{w_{22}} + (1-p_{22}) \cdot x^0) \times (p_{31} \cdot x^{w_{31}} + (1-p_{31}) \cdot x^0) = (0.5x^{0.4} + 0.5x^0) \times (0.8x^{0.9} + 0.2x^0)$. Expand $\mathcal{F}_{Q_4}$, we have, $0.4x^{1.3} + 0.4x^{0.9} + 0.1x^{0.4} + 0.1x^0$. The monomial $0.4x^{1.3}$ can be interpreted as $\Pr\{Q_4.sum = 1.3\} = 0.4$. 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 μ_{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 $\Pr_{maxrs}(Q_x) \geq P_t$), 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 $\Pr_{maxrs}(Q_x) \geq P_t$; 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 $\gamma^*$
   /* initialization                                              */
1  $s \leftarrow O\left(\max\left(\frac{4}{\epsilon^2}\log\frac{2}{\delta_1}, \frac{\log(1-\delta_2)}{\log(1-N^2K^2P_t)}\right)\right)$;
2  $freq[Q_x] \leftarrow 0$ for $\forall Q_x \in C$;
3  $\gamma^* \leftarrow \phi$;
   /* begin sampling                                              */
4  for $i = 1$ to $s$ do
5      sample a possible world $pw$ according to $\mathcal{U}$;
       /* get approximated max range-sum                          */
6      $sum^* \leftarrow \max_{Q_x \in C} Q_x.sum(pw)$;
7      for $Q_x \in C$ do
8          if $Q_x.sum(pw) = sum^*$ then
9              $freq[Q_x] \leftarrow freq[Q_x] + 1$;
10         if $freq[Q_x] > s \cdot P_t$ then
11             $\gamma^* \leftarrow \gamma^* \cup \{Q_x\}$;
12 return $\gamma^*$;
```

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 $freq[Q_x]$, is larger than $s \cdot P_t$, then the region Q_x is selected into the final PMaxRS solution set. That is to say, we use value $\frac{1}{s} freq[Q_x]$ as an approximation of the true PMaxRS probability $\Pr_{maxrs}(Q_x)$.

Our sampling based refinement algorithm, called Verify, is shown in Algorithm 3. The input parameters of Verify are: 1) the set of candidate regions C returned by CandidateGen shown in Algorithm 2; 2) a probabilistic spatial database $\mathcal{U}$, and 3) a probabilistic threshold P_t. Lines 1-3 initialize the sample size s, a PMaxRS solution set γ^* and a frequency list $freq[Q_x]$ which keeps the count that Q_x holds the maximum range-sum through s samplings. Specifically, the sample size s is set to $O\left(\max\left(\frac{4}{\epsilon^2}\log\frac{2}{\delta_1}, \frac{\log(1-\delta_2)}{\log(1-N^2K^2P_t)}\right)\right)$, where ϵ, δ_1, δ_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 $freq[1 \cdots |C|]$. Once $freq[Q_x]$ exceeds $s \cdot P_t$, we add Q_x into the solution set γ^*. The iteration above will be repeated s times and after it terminates, we return γ^* 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 $\max_{Q_x \in C} Q_x.sum(pw)$ 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 $U_i \in \mathcal{U}$, we sample its s instances $u_i^{(1)} \cdots u_i^{(s)}$ based on their probabilities with replacement. Then, $pw_j = \{u_i^{(j)} | i = 1 \cdots N\}$ where $j = 1 \cdots s$, and the probability that pw_j is sampled is $\prod_{i=1}^{N} p_i^{(j)}$. 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 $freq[Q_x]$ 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 $O(sN \log s + s|C|)$, where s is the sample size. Note that, in the worst case, $|C|$ can be $O(N^2K^2)$. However, the experimental results show that by using our pruning technique, most of (more that 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 $s = O\left(\max\left(\frac{4}{\epsilon^2}\log\frac{2}{\delta_1}, \frac{\log(1-\delta_2)}{\log(1-N^2K^2P_t)}\right)\right)$ as shown in Algorithm 3, the probability that the verification of Q_x fails can be bounded within $\min(\delta_1, \delta_2)$, where δ_1 and δ_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 $\frac{1}{s} freq[Q_x]$ to estimate $\Pr_{maxrs}(Q_x)$;
(ii) line 6 uses $\max_{Q_x \in C} Q_x.sum(pw)$ 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 $\Pr_{maxrs}(Q_x)$ and its estimation $\frac{1}{s_1} freq[Q_x]$ 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 $\forall \epsilon, \delta_1 > 0$, it always holds that,*

$$\Pr\left\{\left|\frac{1}{s_1} freq[Q_x] - \Pr_{maxrs}(Q_x)\right| \le \epsilon\right\} \ge 1 - \delta_1, \qquad (7)$$

where $freq[Q_x]$ 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 $\overline{C}$, and a set of sampled possible worlds $PW^s = \{pw_1, pw_2, \cdots, pw_{s_2}\}$, by setting the sample size $s_2 = O\left(\frac{\log(1-\delta_2)}{\log(1-N^2K^2P_t)}\right)$, for $\forall \delta_2 > 0$, it always holds that,*

$$\Pr\left\{\bigwedge_{pw \in PW^s} \bigwedge_{Q_x \in \overline{C}} Q_x.sum(pw) \ne s^*_{pw}\right\} \ge 1 - \delta_2, \qquad (8)$$

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., $s = O\left(\max\left(\frac{4}{\epsilon^2}\log\frac{2}{\delta_1}, \frac{\log(1-\delta_2)}{\log(1-N^2K^2P_t)}\right)\right)$, 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 PMaxRS query and only depends on ϵ and δ_1, which controls the estimation error of the PMaxRS probability. As for the second term, $\frac{\log(1-\delta_2)}{\log(1-N^2K^2P_t)}$ 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 *PMaxRS_Framework*. By slight modification, the *PMaxRS_Framework* can also be used to answer variants of the PMaxRS query, such as PMaxCRS (a circular version of PMaxRS) and top-k 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 *PMaxRS_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 *PMaxRS_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, $\mathbf{E}[Q_x.sum]$, 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:

$$Q_{x^*} = \underset{\forall x \in \mathbb{R}^2}{\arg\max} \sum_{\substack{u_{ij} \in \cup_{i=1}^{N} U_i \\ l_{ij} \in Q_x}} p_{ij} \times w_{ij}. \tag{9}$$

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 *aSB-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 $\frac{4}{\epsilon^2} \log \frac{2}{\delta_1}$. Line 2 initializes an empty hashmap which stores the key-value pair as $(region, count)$, where *region* denotes the possible region in PMaxRS solution set and *count* denotes how many times this region has maximum range-sum among all s 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 *count* value larger than $s \cdot P_t$. 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 $\mathcal{U}$, a region size $a \times b$ and a probability threshold $P_t$
Output: the PMaxRS solution set
1 $s \leftarrow O(\frac{4}{\epsilon^2} \log \frac{2}{\delta_1})$;
2 $cnt\_dict \leftarrow$ empty dictionary;
  /* begin sampling */
3 for $i = 1$ to $s$ do
4     sample a possible world $pw$ according to $\mathcal{U}$;
5     run precise MaxRS on $pw$ with size parameter $a \times b$ and record the result as $Q^*$;
6     if $Q^*$ is key of $cnt\_dict$ then
7         $cnt\_dict[Q^*] \leftarrow cnt\_dict[Q^*] + 1$;
8     else add $(Q^*, 1)$ to $cnt\_dict$ ;
9 return $\{Q | cnt\_dict[Q] \geq s \cdot P_t\}$;
```

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 an 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 database 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 5: Parameter settings.

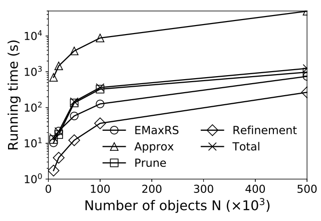
Parameter	Description	Value
N	#uncertain objects	10K, 20K, <u>50K</u>, 100K, 500K
K	#instances per object	2, 3, <u>5</u>, 8, 10
r	length-range of object	5, 10, <u>20</u>, 30, 60
s	sample size	500
P_t	probabilistic threshold	0.01
$a \times b$	size of query region	100×100

5.3 Pruning Power

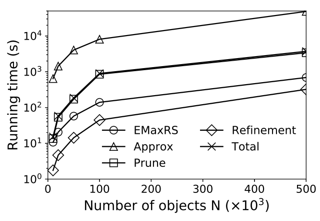
We first report the pruning power of our candidate generation algorithm shown in Algorithm 2. As we discussed in Section 3.2, different selection strategy of the reference region $Q_{\mathcal{A}}$ yields different pruning power. The reason is that, our pruning bound derived in Lemma 2 filters out the regions "worse" than $Q_{\mathcal{A}}$. Thus, an initially good $Q_{\mathcal{A}}$ 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 $Q_{\mathcal{A}}$ 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 $Q_{\mathcal{A}}$ as the region with the highest frequency to be MaxRS solution over k possible worlds; and (3) `EMaxRS`: let $Q_{\mathcal{A}}$ 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.

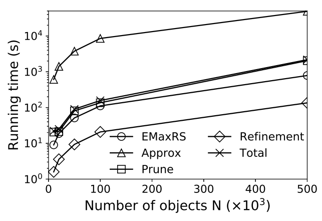
[3]The real datasets are from http://chorochronos.datastories.org.



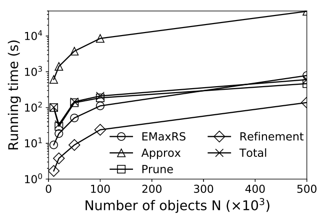
(a) time vs. N (GG)



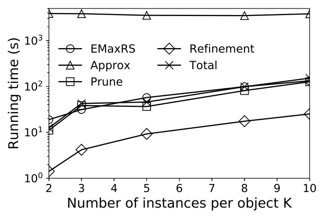
(b) time vs. N (GU)



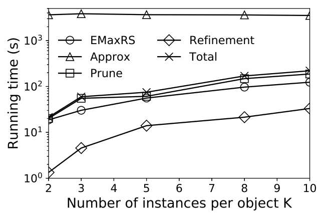
(c) time vs. N (UG)



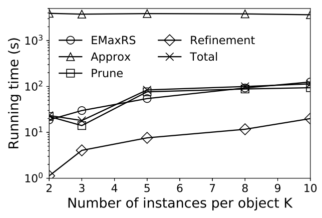
(d) time vs. N (UU)



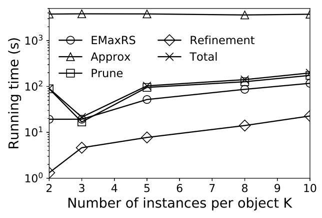
(e) time vs. K (GG)



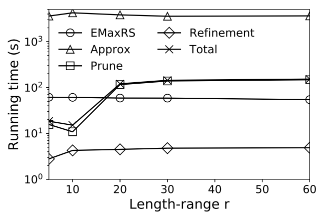
(f) time vs. K (GU)



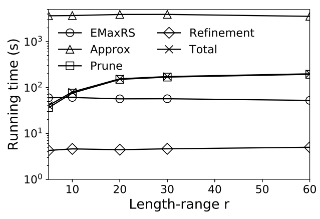
(g) time vs. K (UG)



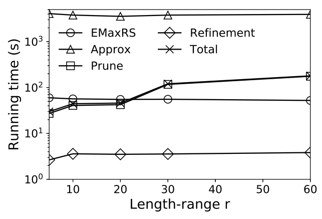
(h) time vs. K (UU)



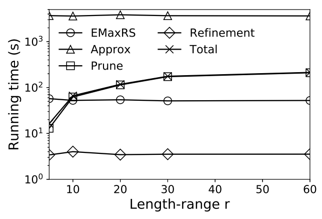
(i) time vs. r (GG)



(j) time vs. r (GU)



(k) time vs. r (UG)



(l) time vs. r (UU)

Figure 6: Experimental results of PMaxRS performance w.r.t. N, K and r.

We can see that the pruning ratios on four datasets are all high. For $\mathsf{k}-\mathsf{Sampling}$ 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 $Q_{\mathcal{A}}$ always obtains the best pruning power among all the three $Q_{\mathcal{A}}$ 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 $\mathsf{k}-\mathsf{Sampling}$ strategy runs much faster, but can still achieve high pruning power. Therefore, in the subsequent experiments, we will use the $\mathsf{k}-\mathsf{Sampling}$ strategy to select the reference region $Q_{\mathcal{A}}$.

Table 6: The pruning results.

	GG	GU	UG	UU
#total candidates	4,150,928	4,150,531	640,670	640,278
MaxPossible	110,452	146,092	40,875	26700
	97.51%	96.48%	93.62%	95.83%
10-sampling	9132	27,808	3,972	4,481
	99.78%	**99.33%**	99.38%	99.30%
EMaxRS	4566	39,843	448	1,729
	99.89%	99.04%	**99.93%**	**99.73%**

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 $x_1, x_2, \cdots,$ and x_{50} from $[0, 10000] \times [0, 10000]$. The absolute error of x_i is defined as $abs\_err_i = |freq_i/s - \Pr_{maxrs}(Q_{x_i})|$, where $freq_i/s$ is the estimated PMaxRS probability $\Pr_{maxrs}(Q_{x_i})$. Then, the total error, $abs\_err = \sum_{i=1}^{50} abs\_err_i$, 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 $s = O\left(\max\left(\frac{4}{\epsilon^2}\log\frac{2}{\delta_1}, \frac{\log(1-\delta_2)}{\log(1-N^2K^2P_t)}\right)\right)$, which guarantees $abs\_err < 5$ with probability 0.9. However, from

Table 7: *PMaxRS_Framework* results on different datasets.

	top-1	top-2	top-3	top-4	top-5	EMaxRS
LB	(5269.5, 1418.7) RS=55, Prob=0.29	(5273.1, 1398.9) RS=53, Prob=0.16	(982.0, 7944.6) RS=53, Prob=0.14	(5268.0, 1401.5) RS=53, Prob=0.10	(1009.3, 7977.0) RS=53, Prob=0.09	(5270.1, 1418.8) RS=55
RR	(198.1, 9410.8) RS=1846, Prob=0.94	(200.8, 9402.3) RS=1844, Prob=0.06	−−	−−	−−	(201.1, 9412.7) RS=1859
GG	(5293.7, 5392.2) RS=104, Prob=0.11	(5300.7, 5390.4) RS=110, Prob=0.11	(5295.5, 5397.5) RS=100, Prob=0.10	(5533.4, 5838.3) RS=101, Prob=0.06	(5527.8, 5838.4) RS=104, Prob=0.06	(5298.5, 5380.0) RS=106
GU	(5303.3, 5382.8) RS=108, Prob=0.09	(5311.0, 5378.2) RS=98, Prob=0.08	(5303.3, 5376.2) RS=101, Prob=0.06	(5309.8, 5387.5) RS=104, Prob=0.05	(5303.3, 5400.0) RS=105, Prob=0.04	(5310.9, 5387.3) RS=104
UG	(2490.0, 7859.5) RS=12, Prob=0.03	(3490.6, 5317.9) RS=13, Prob=0.03	(2515.0, 7879.1) RS=16, Prob=0.03	(2502.1, 7848.1) RS=16, Prob=0.03	(4094.7, 4264.1) RS=15, Prob=0.02	(2502.7, 7848.1) RS=16
UU	(4083.5, 4294.6) RS=16, Prob=0.04	(4250.1, 5908.7) RS=20, Prob=0.04	(6199.5, 4969.4) RS=15, Prob=0.04	(4247.9, 5919.1) RS=15, Prob=0.03	(9362.9, 8960.2) RS=16, Prob=0.03	(4255.2, 5894.4) RS=17

the experimental results, *abs_err* is less than 1 for sample size $s \geq 50$, which is much smaller than 500.

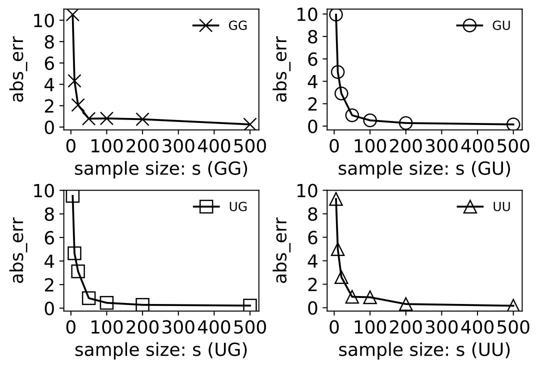


Figure 7: Illustration of the sample error convergence.

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, UU, UG, 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_t as 0.01, top-5 regions usually have significantly high confidences (PMaxRS probabilities). Then, the 6^{th} 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_t. 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.

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, μ 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 correlated case, we need to transform the reference region $Q_{\mathcal{A}}$, 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 *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 $O(nlogn)$, 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 $O(nlog\frac{1}{\epsilon} + nloglogn)$ [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 *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 *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 *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

[1] [n. d.]. Google Map. https://maps.google.com/. Accessed Jan 15, 2018.
[2] [n. d.]. Waze. https://www.waze.com/. Accessed Jan 15, 2018.
[3] Serge Abiteboul, Paris C. Kanellakis, and Gösta Grahne. 1987. On the Representation and Querying of Sets of Possible Worlds. In *SIGMOD Conference*. ACM Press, 34–48.
[4] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB*. Morgan Kaufmann, 487–499.
[5] Daichi Amagata and Takahiro Hara. 2016. Monitoring MaxRS in Spatial Data Streams. In *EDBT*. OpenProceedings.org, 317–328.
[6] Daichi Amagata and Takahiro Hara. 2017. A General Framework for MaxRS and MaxCRS Monitoring in Spatial Data Streams. *ACM Trans. Spatial Algorithms and Systems* 3, 1 (2017), 1:1–1:34.
[7] Carlos D. Barranco, Jesús R. Campaña, and Juan Miguel Medina. 2009. Indexing Fuzzy numerical Data with a B$^+$ Tree for Fast Retrieval Using Necessity-Measured Flexible conditions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 17, Supplement-1 (2009), 1–23.
[8] Douglas Burdick, Prasad Deshpande, T. S. Jayram, Raghu Ramakrishnan, and Shivakumar Vaithyanathan. 2005. OLAP Over Uncertain and Imprecise Data. In *VLDB*. ACM, 970–981.
[9] Zhao Chen, Rui Fu, Ziyuan Zhao, Zheng Liu, Leihao Xia, Lei Chen, Peng Cheng, Caleb Chen Cao, Yongxin Tong, and Chen Jason Zhang. 2014. gMission: A General Spatial Crowdsourcing Platform. *PVLDB* 7, 13 (2014), 1629–1632.
[10] Zitong Chen, Yubao Liu, Raymond Chi-Wing Wong, Jiamin Xiong, Xiuyuan Cheng, and Peihuan Chen. 2015. Rotating MaxRS queries. *Inf. Sci.* 305 (2015), 110–129.
[11] Reynold Cheng, Lei Chen, Jinchuan Chen, and Xike Xie. 2009. Evaluating probability threshold k-nearest-neighbor queries over uncertain data. In *EDBT (ACM International Conference Proceeding Series)*, Vol. 360. ACM, 672–683.
[12] Reynold Cheng, Yuni Xia, Sunil Prabhakar, Rahul Shah, and Jeffrey Scott Vitter. 2004. Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data. In *VLDB*. Morgan Kaufmann, 876–887.
[13] Dong-Wan Choi, Chin-Wan Chung, and Yufei Tao. 2012. A Scalable Algorithm for Maximizing Range Sum in Spatial Databases. *PVLDB* 5, 11 (2012), 1088–1099.
[14] Yuan Shih Chow and Henry Teicher. 2012. *Probability Theory: Independence, Interchangeability, Martingales*. Springer Science & Business Media.
[15] Yang Du, Donghui Zhang, and Tian Xia. 2005. The Optimal-Location Query. In *SSTD (Lecture Notes in Computer Science)*, Vol. 3633. Springer, 163–180.
[16] Parikshit Gopalan, Adam R. Klivans, Raghu Meka, Daniel Stefankovic, Santosh Vempala, and Eric Vigoda. 2011. An FPTAS for #Knapsack and Related Counting Problems. In *FOCS*. IEEE Computer Society, 817–826.
[17] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American statistical association* 58, 301 (1963), 13–30.
[18] Hiroshi Imai and Takao Asano. 1983. Finding the Connected Components and a Maximum Clique of an Intersection Graph of Rectangles in the Plane. *J. Algorithms* 4, 4 (1983), 310–323.
[19] Jon M. Kleinberg, Yuval Rabani, and Éva Tardos. 1997. Allocating Bandwidth for Bursty Connections. In *STOC*. ACM, 664–673.
[20] Jian Li, Barna Saha, and Amol Deshpande. 2009. A Unified Approach to Ranking in Probabilistic Databases. *PVLDB* 2, 1 (2009), 502–513.
[21] Xiang Lian and Lei Chen. 2008. Monochromatic and bichromatic reverse skyline search over uncertain databases. In *SIGMOD Conference*. ACM, 213–226.

[22] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. 2009. WhereNext: a location predictor on trajectory pattern mining. In *KDD*. ACM, 637–646.
[23] Yuki Nakayama, Daichi Amagata, and Takahiro Hara. 2017. Probabilistic MaxRS Queries on Uncertain Data. In *DEXA (1) (Lecture Notes in Computer Science)*, Vol. 10438. Springer, 111–119.
[24] Subhas C Nandy and Bhargab B Bhattacharya. 1995. A Unified Algorithm for Finding Maximum and Minimum Object Enclosing Rectangles and Cuboids. *Computers & Mathematics with Applications* 29, 8 (1995), 45–61.
[25] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. 2007. Probabilistic Skylines on Uncertain Data. In *VLDB*. ACM, 15–26.
[26] Christopher Ré and Dan Suciu. 2009. The trichotomy of HAVING queries on a probabilistic database. *VLDB J.* 18, 5 (2009), 1091–1116.
[27] Mohamed A. Soliman, Ihab F. Ilyas, and Kevin Chen-Chuan Chang. 2007. Top-*k* Query Processing in Uncertain Databases. In *ICDE*. IEEE Computer Society, 896–905.
[28] Yufei Tao, Reynold Cheng, Xiaokui Xiao, Wang Kay Ngai, Ben Kao, and Sunil Prabhakar. 2005. Indexing Multi-Dimensional Uncertain Data with Arbitrary Probability Density Functions. In *VLDB*. ACM, 922–933.
[29] Yufei Tao, Xiaocheng Hu, Dong-Wan Choi, and Chin-Wan Chung. 2013. Approximate MaxRS in Spatial Databases. *PVLDB* 6, 13 (2013), 1546–1557.
[30] Hien To, Gabriel Ghinita, and Cyrus Shahabi. 2014. A Framework for Protecting Worker Location Privacy in Spatial Crowdsourcing. *PVLDB* 7, 10 (2014), 919–930.
[31] Raymond Chi-Wing Wong, M. Tamer Özsu, Philip S. Yu, Ada Wai-Chee Fu, and Lian Liu. 2009. Efficient Method for Maximizing Bichromatic Reverse Nearest Neighbor. *PVLDB* 2, 1 (2009), 1126–1137.
[32] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. 2014. Data Mining with Big Data. *IEEE Trans. Knowl. Data Eng.* 26, 1 (2014), 97–107.
[33] Tian Xia, Donghui Zhang, Evangelos Kanoulas, and Yang Du. 2005. On Computing Top-t Most Influential Spatial Sites. In *VLDB*. ACM, 946–957.
[34] Xiaokui Xiao, Bin Yao, and Feifei Li. 2011. Optimal location queries in road network databases. In *ICDE*. IEEE Computer Society, 804–815.
[35] Man Lung Yiu, Xiangyuan Dai, Nikos Mamoulis, and Michail Vaitis. 2007. Top-k Spatial Preference Queries. In *ICDE*. IEEE Computer Society, 1076–1085.
[36] Wenjie Zhang, Xuemin Lin, Jian Pei, and Ying Zhang. 2008. Managing Uncertain Data: Probabilistic Approaches. In *WAIM*. IEEE Computer Society, 405–412.
[37] Zenan Zhou, Wei Wu, Xiaohui Li, Mong-Li Lee, and Wynne Hsu. 2011. MaxFirst for MaxBRkNN. In *ICDE*. IEEE Computer Society, 828–839.

A PROOF OF LEMMA 1

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

$$\begin{aligned}\Pr_{maxrs}(x) &= \sum_{pw\in PW} \Pr(pw)\cdot\delta(Q_x||pw)\\ &= \sum_{pw\in PW} \Pr(pw)\cdot \mathbf{1}\left(\bigwedge_{\forall x'} Q_x.sum(pw) \geq Q_{x'}.sum(pw)\right)\\ &= \Pr\left\{\bigwedge_{\forall x'} Q_x.sum \geq Q_{x'}.sum\right\},\end{aligned} \tag{10}$$

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

B PROOF OF LEMMA 2

Proof. According to Lemma 1, we have:

$$\begin{aligned}\Pr_{maxrs}(Q_x) &= \Pr\left\{\bigwedge_{\forall x'} Q_x.sum \geq Q_{x'}.sum\right\}\\ &\leq \Pr\{Q_x.sum \geq Q_{\mathcal{A}}.sum\}\\ &= \Pr\{Q_x.sum - Q_{\mathcal{A}}.sum - \mu \geq -\mu\}.\end{aligned} \tag{11}$$

Then, we use *Cantelli's Inequality* [14], that is, for a random variable X and a positive number λ, it holds that:

$$\Pr\{X-\mu_X \geq \lambda\} \leq \frac{\sigma_X^2}{\sigma_X^2+\lambda^2}, \tag{12}$$

where μ_X and σ_X^2 are expectation and variance of X respectively. Combining Eq. (11) with Eq. (12), set $X = Q_x.sum - Q_{\mathcal{A}}.sum$ and $\lambda = -\mu$. Since $-\mu = -\mathbf{E}[Q_x.sum - Q_{\mathcal{A}}.sum] = \mathbf{E}[Q_{\mathcal{A}}.sum] - \mathbf{E}[Q_x.sum]$, if $\mathbf{E}[Q_x.sum] < \mathbf{E}[Q_{\mathcal{A}}.sum]$, then $\lambda = -\mu > 0$. Thus, the condition of *Cantelli's Inequality* is satisfied and we have:

$$\begin{aligned}\Pr_{maxrs}(Q_x) &\leq \Pr\{Q_x.sum - Q_{\mathcal{A}}.sum - \mu \geq -\mu\}\\ &\leq \frac{\sigma^2}{\sigma^2+\mu^2}.\end{aligned} \tag{13}$$

Note that, the condition of *Cantelli's Inequality* cannot be always satisfied. If $\mathbf{E}[Q_x.sum] \geq \mathbf{E}[Q_{\mathcal{A}}.sum]$, $\lambda = -\mu \leq 0$, which breaks the condition of *Cantelli's Inequality*, then the upper bound of $\Pr_{maxrs}(Q_x)$ is set to 1, which is the maximum possible value of a probability. This way, we prove the upper bound of $\Pr_{maxrs}(Q_x)$ 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 $\mu_{Q_x} \geq \mu_{Q_{\mathcal{A}}}$, the upper bound of $\Pr_{maxrs}(Q_x)$ 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 $\mu_{Q_x} < \mu_{Q_{\mathcal{A}}}$, the upper bound of $\Pr_{maxrs}(Q_x)$ is $\frac{\sigma^2}{\sigma^2+\mu^2}$, where $\mu = \mathbf{E}[Q_x.sum - Q_{\mathcal{A}}.sum]$ and $\sigma^2 = \mathbf{Var}[Q_x.sum - Q_{\mathcal{A}}.sum]$. Since $Q_x.sum$ and $Q_{\mathcal{A}}.sum$ are independent, σ^2 can be decomposed as $\sigma_{Q_x}^2 + \sigma_{Q_{\mathcal{A}}}^2$. Then, we have:

$$\sigma_{Q_x}^2 + \sigma_{Q_{\mathcal{A}}}^2 > \frac{P_t\cdot(\mu_{Q_x}-\mu_{Q_{\mathcal{A}}})^2}{1-P_t}. \tag{14}$$

Since $\sigma_{Q_x}^2 = \mathbf{MS} - \mu_{Q_x}^2$, $\mathbf{MS} - \mu_{Q_x}^2 + \sigma_{Q_{\mathcal{A}}}^2 > \frac{P_t\cdot(\mu_{Q_x}-\mu_{Q_{\mathcal{A}}})^2}{1-P_t}$. Simplify it, we have:

$$\frac{1}{1-P_t}\mu_{Q_x}^2 - \frac{2P_t\cdot\mu_{Q_{\mathcal{A}}}}{1-P_t}\mu_{Q_x} + \frac{P_t}{1-P_t}\mu_{Q_{\mathcal{A}}}^2 - \mathbf{MS} - \sigma_{Q_{\mathcal{A}}}^2 < 0. \tag{15}$$

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

$$\begin{aligned}\Delta &= \left(\frac{2P_t\cdot\mu_{Q_{\mathcal{A}}}}{1-P_t}\right)^2 - \frac{4}{1-P_t}\left(\frac{P_t}{1-P_t}\mu_{Q_{\mathcal{A}}}^2 - \mathbf{MS} - \sigma_{Q_{\mathcal{A}}}^2\right)\\ &= \frac{4}{1-P_t}(\mathbf{MS} + \sigma_{Q_{\mathcal{A}}}^2 - P_t\cdot\mu_{Q_{\mathcal{A}}}^2).\end{aligned}$$

To satisfy the inequality shown in Eq. (15) with the constraint $\mu_{Q_x} \in [0, \mu_{Q_{\mathcal{A}}}]$, there should be $\Delta > 0$ and $\mu_{Q_x} \in [\max(0,\mu_0^-), \min(\mu_{Q_{\mathcal{A}}}, \mu_0^+)]$, where μ_0^+ and μ_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 $Q_{\mathcal{A}}$ covers more uncertain objects than $Q_{\mathcal{A}}^{\dagger}$, by following the similar procedures of proving Lemma 2, it holds that:

$$\begin{aligned}\Pr_{maxrs}(Q_x) &\leq \Pr\{Q_x.sum \geq Q_{\mathcal{A}}.sum\}\\ &\leq \Pr\{Q_x.sum \geq Q_{\mathcal{A}}^{\dagger}.sum\}\\ &\leq \frac{\sigma^{\dagger 2}}{\sigma^{\dagger 2}+\mu^{\dagger 2}}.\end{aligned} \tag{16}$$

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 $\Pr(Q_x.sum \geq Q_{x'}.sum)$, where $Q_{x'}$ is any region different from Q_x, is a special case of calculating $\Pr_{maxrs}(Q_x)$. And similarly, $\Pr(Q_x.sum \geq L)$, where L is an arbitrary positive constant, is also a special case of the probability $\Pr(Q_x.sum \geq Q_{x'}.sum)$. Thus, $\Pr(Q_x.sum \geq L)$ is a special case of $\Pr_{maxrs}(Q_x)$. Then, according to [19, 26], the evaluation of $\Pr(Q_x.sum \geq L)$ 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 $\Pr_{maxrs}(Q_x)$ 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 $i = 1, 2, \cdots$, and $|PW|$, 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 $\Pr_{maxrs}(Q_x) = \mathbf{E}[X]$.

Table 8: Distribution of random variable X.

X	1	$\cdots$	1	0	$\cdots$	0
Pr	$\Pr(pw_1)$	$\cdots$	$\Pr(pw_k)$	$\Pr(pw_{k+1})$	$\cdots$	$\Pr(pw_{\lvert PW\rvert})$

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

$$\Pr\left\{\frac{1}{s_1}\sum_{i=1}^{s_1} X_i \geq \mathbf{E}[X] + \epsilon\right\} \leq e^{-D(\mathbf{E}[X]+\epsilon||\mathbf{E}[X])}, \tag{17}$$

where $D(\cdot||\cdot)$ is *K-L Divergence*. Use bound: $D((1+x)p||p) \geq \frac{1}{4}x^2 p$ for $x \in [-\frac{1}{2}, \frac{1}{2}]$, we have:

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

Similarly, we have,

$$\Pr\left\{\frac{1}{s_1}\sum_{i=1}^{s_1} X_i \leq \mathbf{E}[X] - \epsilon\right\} \leq e^{-\frac{\epsilon^2}{4}\cdot s_1}. \tag{19}$$

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

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

By substituting s_1 with $\frac{4}{\epsilon^2}log\frac{2}{\delta_1}$, we finally have $\Pr\{|\frac{1}{s_1}\sum_{i=1}^{s_1} X_i - \mathbf{E}[X]| \leq \epsilon\} \geq 1 - \delta_1$, which is equivalent to $\Pr\{|\frac{1}{s_1}freq[Q_x] - \Pr_{maxrs}(Q_x)| \leq \epsilon\} \geq 1 - \delta_1$. □

G PROOF OF THEOREM 3

Proof. For any non-candidate region $Q_x \in \overline{C}$, since Q_x is filtered out, there must be $\Pr_{mxars}(Q_x) < P_t$. Combining Eq. (2), for a sampled possible world $pw_i \in PW^s$, we have:

$$\begin{aligned}\Pr\left\{Q_x.sum(pw_i) = s^*_{pw_i}\right\} &= \delta(Q_x||pw_i)\cdot\Pr(pw_i)\\ &\leq \sum_{pw\in PW}\delta(Q_x||pw)\cdot\Pr(pw)\\ &\leq \Pr_{maxrs}(Q_x) < P_t.\end{aligned} \tag{21}$$

By using the union bound of probability, we have,

$$\Pr\left\{\bigwedge_{Q_x\in\overline{C}} Q_x.sum(pw_i) \neq s^*_{pw_i}\right\}$$

$$=1 - \Pr\left\{\bigvee_{Q_x\in\overline{C}} Q_x.sum(pw_i) = s^*_{pw_i}\right\} \tag{22a}$$

$$\geq 1 - \sum_{Q_x\in\overline{C}}\Pr\left\{Q_x.sum(pw_i) = s^*_{pw_i}\right\} \tag{22b}$$

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

Note that, from Eq. (22b) to Eq. (22c), we utilize the fact that there are at most $\frac{N\cdot K\cdot(NK-1)}{2}$ possible candidate regions, where K is the average number of instances per uncertain object.

Then, since possible worlds $pw \in PW^s$ are independently sampled, we have,

$$\Pr\left\{\bigwedge_{pw\in PW^s}\bigwedge_{Q\in\overline{C}} Q.sum(pw) \neq s^*_{pw}\right\} \geq (1-(NK)^2P_t)^{s_2}$$

Let $s_2 = O\left(\frac{\log(1-\delta_2)}{\log(1-N^2K^2P_t)}\right)$, 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 *PMaxrs_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.