Range-Max Queries on Uncertain Data*

Pankaj K. Agarwal† Nirman Kumar‡ Stavros Sintos§ Subhash Suri¶

ABSTRACT

Let P be a set of n uncertain points in $\mathbb{R}^d$, where each point $p_i \in P$ is associated with a real value v_i and a probability $\alpha_i \in (0, 1]$ of existence, i.e., each p_i exists with an independent probability α_i. We present algorithms for building an index on P so that for a d-dimensional query rectangle ρ, the expected maximum value or the most-likely maximum value in ρ can be computed quickly. The specific contributions of our paper include the following: (i) The first index of sub-quadratic size to achieve a sub-linear query time in any dimension $d \geq 1$. It also provides a trade-off between query time and size of the index. (ii) A conditional lower bound for the most-likely range-max queries, based on the conjectured hardness of the set-intersection problem, which suggests that in the worst case the product (query time)^2 $\times$ (index size) is $\Omega(\frac{n^2}{\text{polylog}(n)})$. (iii) A linear-size index for estimating the expected range-max value within approximation factor $1/2$ in $O(\log^c n)$ time, for some constant $c > 0$; that is, if the expected maximum value is μ then the query procedure returns a value μ' with $\mu/2 \leq \mu' \leq \mu$. (iv) Extensions of our algorithm to more general uncertainty models and for computing the top-k values of the range-max.

1. INTRODUCTION

Query-driven data management is an important function in most database systems, and range query is a common tool for summarizing information on the objects lying in a query range. In recent years, motivated by applications in sensor networks, data cleaning, data integration, pervasive computing and scientific data analysis, there has been a growing interest in managing *uncertain data* [6, 14]. In these settings, uncertainty is typically captured using stochastic data models, and summarizing or querying data requires computing *statistics* about the probabilistic behavior of the underlying data.

In this paper we study the *range-max* query on uncertain data—return statistics on the maximum value of the data inside a query range. Formally, we consider database records of the form (p, v, α), where p is a *point* in $\mathbb{R}^d$ (attribute values of the record) with $d \geq 1$ a constant, along with a positive *scalar value* $v \in \mathbb{R}^+$ and a probability $\alpha \in (0, 1]$. We refer to this simple model as the *existence-uncertainty* model. The value v is the metric of interest in our range queries and the probability α reflects our confidence in this record. Given a collection of n such records, our goal is to construct an index to answer queries of the following form efficiently: report the *expected maximum value* (the EM problem) or the *most likely maximum value* (MLM problem) of the records whose attribute values (points) lie in a d-dimensional orthogonal query rectangle ρ. We use P to denote the set of points corresponding to the input records. Throughout the paper, we assume the real-RAM model of computation, which counts word-level operations instead of bits: specifically, each memory cell can store an arbitrary real number and any arithmetic or relational operation between two operands takes $O(1)$ time.

The max is a special case of the widely studied top-k summary of data (i.e. $k = 1$) but is also a popular

*Work by Agarwal and Sintos is supported by NSF under grants CCF-11-61359, CCF-15-13816, CCF-15-46392, and IIS-14-08846, by ARO grant W911NF-15-1-0408, and by Grant 2012/229 from the U.S.-Israel Binational Science Foundation. Work by Suri and Kumar is supported by NSF under grants CCF-11-61495 and CCF-15-25817.

†Department of Computer Science, Duke University, Durham, NC 27708-0129, USA; pankaj@cs.duke.edu.

‡Department of Computer Science, University of California, Santa Barbara, CA 93106, USA; nirman@cs.ucsb.edu.

§Department of Computer Science, Duke University, Durham, NC 27708-0129, USA; ssintos@cs.duke.edu.

¶Department of Computer Science, University of California, Santa Barbara, CA 93106, USA; suri@cs.ucsb.edu.

PODS'16, June 26-July 01, 2016, San Francisco, CA, USA

© 2016 ACM. ISBN 978-1-4503-4191-2/16/06…$15.00

DOI: http://dx.doi.org/10.1145/2902251.2902281

aggregation operator in its own right. For example, the suprema of random processes (a collection of random variables arising from given distributions) are widely studied in random matrix theory, control of empirical processes in statistics and machine learning, random optimization problems, and Banach spaces probabilities. The reason for this keen interest can be understood from the following hypothetical but representative application. The points might represent a set of geographical locations (cities) $\{p_1, p_2, \ldots, p_n\}$, each associated with a probability α_i of being struck by a natural disaster (flood, earthquake, fire) during next year, and v_i a measure of the *cost* (damage) incurred at that location due to that disaster.[1] In this case, a range query asks for the expected value of the maximum damage suffered within the range. Similarly, an insurance company may associate probabilities of financial claims with various entities, and need to analyze the profile of its maximum loss portfolio. Indeed, in natural disasters such as earthquakes or flooding, the impact is highly *non-linear*—even hundreds of small quakes hardly cause serious financial or social harm, but a single large one can be catastrophic. Thus, it is far more important to be able to carry out analysis on the profile of the *maximum values*, and not on simpler aggregates such as sum or average.

Range-max queries on uncertain data are also relevant when dealing with spatially distributed noisy data sources (e.g. sensors) where unusually high measurements might be cause for concern, but only if they deviate from the norm. A probabilistic profile of the expected max in a range can serve as the benchmark for deciding when a sensor measurement is abnormal.

We point out that, given a query rectangle ρ, the expected value of *range-SUM*, namely, $\sum_{p_i \in \mathsf{P} \cap \rho} \alpha_i v_i$ is easy to compute: we simply assign each point p_i a *weight* of $\alpha_i v_i$ and compute the weighted range sum using traditional (non-uncertain) techniques. In contrast, computing the expected value of *range-max*, our EM problem, seems much harder and it is not clear how to use any existing range-aggregation index to answer this query. The difficulty arises in part because the EM problem is not *range decomposable*, i.e., if P_1, P_2 is a partition of P then for a query rectangle ρ, the EM value of $\mathsf{P} \cap \rho$ cannot be quickly computed from the EM values of $\mathsf{P}_1 \cap \rho$ and $\mathsf{P}_2 \cap \rho$. The interaction of probabilities rules out traditional tree-style range-searching indexes where the value at each node is inferred simply from values at its children. On the positive side, while the probability distribution of range-SUM can take exponentially many possible values, the probability distribution of range-max only has linear size. Therefore, computing statistics on the distribution of range-max (e.g. most likely max) might be easier than a similar statistics on range-SUM; the latter is known to be $\#P$-hard [19].

[1]More generally, each point can be associated with not just a single pair (value, probability) but an entire distribution. For the ease of exposition, we initially focus on the single value case, but remark on how to generalize our results to distributional settings later.

For both the EM and MLM queries in 1-dimension, an index structure with $O(n^2)$ size and $O(\log n)$ query time is straightforward: we can precompute answers for each of the $O(n^2)$ *combinatorially distinct* intervals defined by pairs of input points. The interesting question is whether these queries can be answered in sublinear time using subquadratic space. In this paper, we answer this question affirmatively.

Our results. Our main results are shown in Table 1, and can be summarized as follows:

(A) We design the first sub-quadratic size index that achieves a sub-linear query time in any dimension $d \geq 1$ for both EM and MLM problems. For any $t \in [0, 1]$, the index answers a query in $O(n^{1-t} + \log n)$ time, has size $O(n^{(2d-1)t+1})$, and takes $O(n^{(2d-1)t+1} \log n)$ time to build. The tunable parameter t gives the index a continuum of trade-offs between query time and its size. In particular, for $d = 1$ the index can achieve a query time of $O(\sqrt{n})$ with $O(n^{3/2})$ size and $O(n^{3/2} \log n)$ preprocessing. (Section 2)

(B) Our above result raises a natural question: is there a near-linear-size index that can answer an EM or MLM query in polylog(n) time. Although no such lower bound is known, we show that such an index is unlikely. In particular, we prove a conditional lower bound for the most-likely range-max queries, based on the conjectured hardness of the following set-intersection problem: Preprocess a family $S_1, \ldots, S_m$ of sets so that for a query pair $i, j \leq m$, one can quickly determine whether $S_i \cap S_j = \emptyset$. It is conjectured that the size of any data structure with query time $t_q = \Omega(\log n)$ is $\Omega((\frac{n}{t_q \operatorname{polylog}(n)})^2)$, where $n = \sum_i |S_i|$. We reduce the MLM problem for $d \geq 2$ to the set-intersection problem. (Section 3)

Since the EM and MLM problems seem hard in the worst-case, we consider approximation algorithms and algorithms under some assumptions on the input, as discussed below.

(C) Suppose the values of input points are chosen according to a random permutation model (RPM), i.e., first a set of values are chosen and then they are assigned in a random order to the input points (see Section 4). We present an index of size $O(n \log^{d+1} n)$ that can answer a MLM query in expected time $O(\log^{d+3} n)$. (Section 4)

(D) We design a linear-size index for estimating the expected maximum value (denoted by $\mu(\mathsf{P} \cap \rho)$), for a query rectangle ρ, within approximation factor $1/2$ in $O(\log^{d+1} n)$ time; that is, the query procedure returns a value μ' such that

$$\frac{1}{2}\mu(\mathsf{P} \cap \rho) \leq \mu' \leq \mu(\mathsf{P} \cap \rho).$$

(Section 5)

Finally we extend our algorithms in two directions:

(E) First, we extend our algorithms to the *location-uncertainty* model, in which the location of each point

Model	Algorithm	EM		MLM	
		Space	Query	Space	Query
Existence	Exact	$O\left(n^{(2d-1)t+1}\right)$	$O\left(n^{1-t}+\log n\right)$	$O\left(n^{(2d-1)t+1}\right)$	$O\left(n^{1-t}+\log n\right)$
				$\Omega\left(\left(\frac{n}{t_q \operatorname{polylog}(n)}\right)^2\right)$	t_q
	Approx.	$O\left(n\log^d n\right)$	$O\left(\log^{d+1} n\right)$		
	RPM			$O\left(n\log^{d+1} n\right)$	$O\left(\log^{d+3} n\right)$
Location	Exact	$O\left(n^{(2d-1)t+1}f^{2d}\right)$	$O\left(n^{1-t}+\log(nf)\right)$	$O\left(n^{(2d-1)t+1}f^{2d}\right)$	$O\left(n^{1-t}+\log(nf)\right)$
		$\Omega\left(\left(\frac{n}{t_q \operatorname{polylog}(n)}\right)^2\right)$	t_q	$\Omega\left(\left(\frac{n}{t_q \operatorname{polylog}(n)}\right)^2\right)$	t_q
	Approx.	$O\left(nf\log^d(nf)\right)$	$O\left(\log^{d+1}(nf)\right)$		
	RPM			$O\left(nf^{2d}\log^{2d+1}(nf)\right)$	$O\left(\log^{2d+3}(nf)\right)$

Table 1. Summary of results. Lower bounds are conditional to the set-intersection conjecture and hold for $d \geq 2$; RPM is the random permutation model; and t_q is the query time.

is described as a probability distribution over a set of f discrete locations. We propose generalizations of the subquadratic index in (A) to compute the EM or the MLM in sublinear time and generalizations of (D) to compute an 1/2-approximation of the expected maximum value in polylog(n) time. We also provide a conditional lower bound for the EM problem in location distributions. Second, we provide an indexing scheme for returning k records with the highest probabilities of being the maximum. (Sections 2, 6)

Our paper makes significant progress in advancing the state of art for computing range-max statistics over uncertain data. Despite the fundamental role and practical applications of the max statistic, and top-k summaries more generally, prior to our work no sublinear query scheme with subquadratic storage was known for either the EM or the MLM problem. In achieving sublinear query time using subquadratic space, our proposed algorithms break a conceptual barrier, and introduce novel indexing techniques both for exact and approximate range-max queries. In the exact setting, the idea of first partitioning data by value (instead of range attributes), and then applying range searching within each group turns out to be a key insight for range-max queries. In the approximation setting, the use of Prophet inequality for range queries is novel. Finally, our hardness proof adapts a construction originally designed for deterministic range queries, and applies it to derive complexity lower bounds for uncertain data ranges.

Related work. There is extensive work on managing, querying, and analyzing uncertain data. See the book [6] and the survey paper [14] for an overview of known results on this topic.

In the context of query processing, early work focused on top-k queries over uncertain data. Numerous algorithms under different semantics of top-k queries have been proposed; see [11, 16, 17, 22, 27, 30] and references therein. In another line of work, Jayram *et al.* [18] describe algorithms for computing aggregation statistics on uncertain data in the streaming model. Cormode *et al.* [13] used the expected rank of the uncertain entries across all possible worlds to compute the ranking of the data. All these papers compute the desired statistics over the entire data, and do not consider query ranges. Extending these results to top-k (even for $k = 1$) or aggregation statistics inside a query range seems particularly challenging, partly because the problem is not range decomposable, as mentioned above.

A few algorithms have been proposed for answering range-reporting queries on uncertain data [28, 31]. Cheng *et al.* [12] considered the 1D range-reporting query when the location of each input point is given as a uniform distribution and the goal is to report all points that lie inside a query range with probability at least τ, for some fixed $\tau \in [0, 1]$. They proposed an index of $O(n\tau^{-1})$ size with $O(\tau^{-1}\log n + k)$, where k is is the output size. Note that both the size and the query time depend on τ. Tao *et al.* [28] considered the problem in higher dimensions, and gave indexes based on space partitioning heuristics. In the worst case, their procedure visits all points.

Agarwal *et al.* [4] described an index for 1D range reporting when the location of each point is given as a piecewise-constant pdf. For a fixed τ, they described an index of linear size that can report in $O(\log n + k)$ time all k points lying inside a query interval with probability at least τ. If τ is part of the query, then the query time is $O(\log^3 n + k)$ and the size of the index is $O(n \log^2 n)$. They also presented a near-linear-size index that can handle more general probability distributions for input points and answer queries in $O(\log n + k)$ time. Recently Li and Wang [21] extended the approach in [4] to return the t points, for a given t (instead of all points), that lie in the query interval with the highest probability. Singh *et al.* [26] presented some heuristic solutions for querying uncertain data that are categorical. All these results are customized for range-*reporting* queries, and they require linear time in the worst case for computing range-aggregate statistics.

As mentioned above, computing the expected value of range-SUM for a query rectangle ρ can be reduced to computing range-SUM on data without uncertainty. However no efficient index is known for computing the most-likely value of range-SUM, or approximating the distribution of range-SUM (unless we make some assumptions on the probabilities of existence). The only result

in this direction is by Abdullah *et al.* [1] who proved the existence of small-size coresets for some range-SUM problems on uncertain data.

The only paper on range top-k we are aware of is [29], which studies the problem for $d = 1$. For a query interval I and a parameter $\tau \in [0, 1]$, the data structure returns points that lie in I and their rank of being at most k is at least τ. They present some pruning techniques to reduce the query time, but the worst-case query time is super linear.

Finally we note that there is also some work on nearest neighbor queries on uncertain data [3, 5, 8, 9] but they cannot be adapted to range-max queries.

2. EXACT ALGORITHMS

Let $\mathsf{P} = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$ be a set of n uncertain points where each p_i is associated with a value $v_i \in \mathbb{R}^+$ and an independent probability $\alpha_i \in (0, 1]$ of its existence; multiple points may have the same value. Throughout, we assume that dimension d is a fixed constant. We present indexing schemes for computing the expected or the most-likely maximum value among the points lying in a query orthogonal rectangle in $\mathbb{R}^d$. We begin with some definitions, and then describe our index structure.

Let $X = \{x_1, \ldots, x_m\}$, for $m \leq n$, be the set of (distinct) values associated with the points in P. (Recall that the values of the points need not be all distinct.) Given a subset of points $\mathsf{R} \subseteq \mathsf{P}$, we define $\pi(\mathsf{R})$ as the probability that *none* of its points are present. That is,

$$\pi(\mathsf{R}) = \prod_{p_i \in \mathsf{R}} (1 - \alpha_i),$$

where we assume $\pi(\mathsf{R}) = 1$ if $\mathsf{R} = \emptyset$. For any value $x \in X$, let $\alpha(x, \mathsf{R})$ denote the probability that some point with associated value x exists:

$$\begin{aligned} \alpha(x, \mathsf{R}) &= 1 - \pi\left(\{p_i \in \mathsf{R} \mid v_i = x\}\right) \\ &= 1 - \prod_{p_i \in \mathsf{R}: v_i = x} (1 - \alpha_i), \end{aligned}$$

with the convention that $\alpha(x, \mathsf{R}) = 0$ if there is no point with value x in R; in particular, $\alpha(x, \mathsf{R}) = 0$ if $\mathsf{R} = \emptyset$. Define $\beta(x, \mathsf{R})$ as the probability that no point with value larger than x is present in R:

$$\beta(x, \mathsf{R}) = \pi\left(\{p_i \in \mathsf{R} \mid v_i > x\}\right).$$

It follows that the probability of x being the maximum value of a point in R, denoted $\gamma(x, \mathsf{R})$, can be written as

$$\gamma(x, \mathsf{R}) = \alpha(x, \mathsf{R})\beta(x, \mathsf{R}).$$

Note that $\gamma(x, \mathsf{R}) = 0$ if $\mathsf{R} = \emptyset$ or x is not associated with any point of R. If $\mu(\mathsf{R})$ denotes the expected maximum (EM) value in R, then we have

$$\mu(\mathsf{R}) = \sum_{x \in X} x\gamma(x, \mathsf{R}).$$

By our convention, $\mu(\mathsf{R}) = 0$ if $\mathsf{R} = \emptyset$. The most likely maximum (MLM) value, denoted by $\lambda(\mathsf{R})$, is

$$\lambda(\mathsf{R}) = \operatorname*{argmax}_{x \in X} \gamma(x, \mathsf{R}).$$

If $\mathsf{R} = \emptyset$, $\lambda(\mathsf{R})$ is undefined.

We describe our index structure in detail for computing EM for $d = 1$, and then sketch its extension to higher dimensions and to computing MLM.

Indexing scheme. We fix a parameter $t \in [0, 1]$, and partition the input P into $k \leq 2\lceil n^{1-t} \rceil$ subsets $\mathsf{P}_1, \ldots, \mathsf{P}_k$ such that (i) all points in P_i have strictly higher values than those in P_{i+1}, and (ii) either $|\mathsf{P}_i| \leq 2n^t$ or all points in P_i have the same value.

We call P_i *uniform* if all of its points have the same value, and non-uniform otherwise. If P_i is uniform, then we compute and store the value $\pi(\mathsf{P}_i \cap (-\infty, p_j])$, for all $p_j \in \mathsf{P}_i$. Otherwise (for non-uniform P_i) we compute and store $\pi(\mathsf{P}_i \cap [a, b])$ and $\mu(\mathsf{P}_i \cap [a, b])$, for all pairs $a, b \in \mathsf{P}_i$ with $a < b$.

We also sort each P_i (by position) and store the sorted lists $\mathsf{P}_1, \ldots \mathsf{P}_k$ so that for any $q \in \mathbb{R}$ its predecessor and successor in each P_i, denoted by $pred(q, \mathsf{P}_i)$ and $succ(q, \mathsf{P}_i)$ respectively, can be computed efficiently. We use the so-called fractional-cascading scheme [10] to expedite this process, which augments each sequence P_i by adding some points to P_i and stores the augmented list $\mathsf{P}_i^\star$. More precisely, starting with the bottom-most set $\mathsf{P}_k^\star = \mathsf{P}_k$, we construct the remaining augmented lists as follows. Suppose $\mathsf{P}_i^\star = \langle a_1, a_2, a_3, \ldots, a_{n_i} \rangle$, and let $\frac{1}{2}\mathsf{P}_i^\star = \langle a_2, a_4, a_6, \ldots \rangle$ be the subsequence of $\mathsf{P}_i^\star$ consisting of every other item. Then, we define $\mathsf{P}_{i-1}^\star = \mathsf{P}_{i-1} \cup \frac{1}{2}\mathsf{P}_i^\star$, and with each item $a \in \mathsf{P}_i^\star$, we store the indices of $pred(a, \mathsf{P}_i)$, $succ(a, \mathsf{P}_i)$ and of $succ(a, \mathsf{P}_{i+1}^\star)$ in $\mathsf{P}_{i+1}^\star$. The total size of all the augmented list is $\sum_{i=1}^k |\mathsf{P}_i^\star| \leq 2n$. With these augmented lists, we can find $pred(q, \mathsf{P}_i)$ and $succ(q, \mathsf{P}_i)$ for any $q \in \mathbb{R}$ in all the set P_i, $i = 1, 2, \ldots, k$, in total time $O(\log n + k)$, improving the obvious bound of $O(k \log n)$. See [10] for details. This completes the description of the index.

The index needs $O(|\mathsf{P}_i|^2) = O(n^{2t})$ space when P_i is non-uniform, and $O(|\mathsf{P}_i|)$ space otherwise. Since $\sum_{i=1}^k |\mathsf{P}_i| = n$, the total space for the index is $O(n) + O(n^{1-t}n^{2t}) = O(n^{1+t})$. We now discuss how to answer a query using this index, and then discuss its construction details.

Query procedure. Given a query interval $I = [a, b]$, we wish to compute $\mu(\mathsf{P} \cap I)$ using the index. For $j = 1, \ldots, k$, let $a_j = succ(a, \mathsf{P}_j)$ and $b_j = pred(b, \mathsf{P}_j)$. Define $I_j = [a_j, b_j]$ as the projection of the query range onto the subset P_j, where we assume $I_j = \emptyset$ if $a_j > b_j$, see Figure 1 for an example. We use the shorthand notation $\mathsf{P}_{<j} = \bigcup_{l=0}^{j-1} \mathsf{P}_l$ to denote the union of all subsets with index smaller than j, with the convention that $\mathsf{P}_{<1} = \emptyset$. The following lemma explains our main idea for computing the expected max value $\mu(\mathsf{P} \cap I)$.

Lemma 2.1 *The partition $\mathsf{P}_1, \ldots, \mathsf{P}_k$ satisfies the following properties:*

(i) $\mu(I \cap \mathsf{P}) = \sum_{j=1}^{k} \pi(I \cap \mathsf{P}_{<j})\mu(I_j \cap \mathsf{P}_j)$.

(ii) For $j \geq 1$, $\pi(I \cap \mathsf{P}_{<j}) = \pi(I \cap \mathsf{P}_{<j-1})\pi(I_{j-1} \cap \mathsf{P}_{j-1})$.

PROOF. To prove (i), let $X_j \subseteq X$ be the set of distinct values among the points of P_j. Then the following holds:

$$\begin{aligned}\mu(I \cap \mathsf{P}) &= \sum_{x \in X} x\alpha(x, I \cap \mathsf{P})\beta(x, I \cap \mathsf{P}) \\ &= \sum_{j=1}^{k} \sum_{x \in X_j} x\alpha(x, I \cap \mathsf{P})\beta(x, I \cap \mathsf{P})\end{aligned}$$

The values in X_{j-1} are all larger than those in X_j and x belongs to at most one such X_j, therefore $\alpha(x, I \cap \mathsf{P}) = \alpha(x, I \cap \mathsf{P}_j)$, and

$$\begin{aligned}\beta(x, I \cap \mathsf{P}) &= \pi(I \cap \mathsf{P}_{<j})\pi(I \cap \{p_i \in \mathsf{P}_j \mid v_i > x\}) \\ &= \pi(I \cap \mathsf{P}_{<j})\beta(x, \mathsf{P}_j \cap I).\end{aligned}$$

We also have $\alpha(x, I \cap \mathsf{P}_j) = \alpha(x, I_j \cap \mathsf{P}_j)$ and $\beta(x, I \cap \mathsf{P}_j) = \beta(x, I_j \cap \mathsf{P}_j)$, which follows because $\mathsf{P}_j \cap (I \backslash I_j) = \emptyset$, by definition. Thus, we can conclude that

$$\begin{aligned}\mu(I \cap \mathsf{P}) &= \sum_{j=1}^{k} \sum_{x \in X_j} x\pi(I \cap \mathsf{P}_{<j})\alpha(x, I_j \cap \mathsf{P}_j)\beta(x, I_j \cap \mathsf{P}_j) \\ &= \sum_{j=1}^{k} \pi(I \cap \mathsf{P}_{<j})\mu(x, I_j \cap \mathsf{P}_j).\end{aligned}$$

The proof of (ii) follows from the fact that $\mathsf{P}_{<j} = \mathsf{P}_{<j-1} \cup \mathsf{P}_{j-1}$ and $\mathsf{P}_{<j-1} \cap \mathsf{P}_{j-1} = \emptyset$. □

If P_j is non-uniform, then the index stores the values of $\mu(I_j \cap \mathsf{P}_j)$ and $\pi(I_j \cap \mathsf{P}_j)$. For a uniform P_j, we observe that $\pi(I_j \cap \mathsf{P}_j) = \pi(\mathsf{P}_j \cap (-\infty, b_j])/\pi(\mathsf{P}_j \cap (-\infty, a_j))$ and $\mu(I_j \cap \mathsf{P}_j) = x(1 - \pi(I_j \cap \mathsf{P}_j))$. Since we store $\pi(\mathsf{P}_j \cap (-\infty, p])$ for all $p \in \mathsf{P}_j$, the values $\pi(I_j \cap \mathsf{P}_j)$ and $\mu(I_j \cap \mathsf{P}_j)$ can be computed in $O(1)$ time. The pseudo-code in Algorithm 1 below describes the query procedure and Figure 1 shows the intervals in each group given the query interval I.

Algorithm 1. EXACT_EM_QUERY

Input: $I = [a, b]$
Output: $\mu(I \cap \mathsf{P})$

```
1: for j = 1 to k do
2:     a_j = succ(a, P_j),    b_j = pred(b, P_j)
3: end for
4: EM = 0,    p = 1
5: for j = 1 to k do
6:     EM = EM + p · μ(I_j ∩ P_j)
7:     p = p · π(I_j ∩ P_j)
8: end for
9: return EM
```

We note that a_j and b_j can be found for all $j = 1, 2, \ldots, k$, in $O(\log n + k)$ time using fractional-cascading. We spend $O(1)$ time in each iteration of the second for loop, so the total query time is $O(\log n + k) = O(\log n + n^{1-t})$.

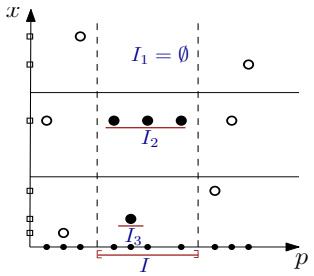

Figure 1. Points (x-axis), values (y-axis), and a partition of P into 3 sets; $\{(p_i, v_i) \mid 1 \leq i \leq n\}$ are drawn as $2D$ points.

Preprocessing. We now describe how to build the index structure. Given a subset $S \subseteq \mathsf{P}$, let $S[x] = \{p_i \in S \mid v_i = x\}$ be the set of points in S that have value x. We sort the set of distinct values X in descending order and construct the partition of P by scanning the elements of X in sorted order. Suppose the sets $\mathsf{P}_1, \ldots, \mathsf{P}_{j-1}$ have been constructed, we are building the set P_j, and $x_i \in X$ is the next value in the sorted sequence. If either $|\mathsf{P}_j| \geq n^t$ or $|\mathsf{P}[x_i]| \geq n^t$, we close P_j, and start the new subset P_{j+1} by adding $\mathsf{P}[x_i]$ to it. Otherwise we have both $|\mathsf{P}_j| < n^t$ and $|\mathsf{P}[x_i]| < n^t$, in which case we add $\mathsf{P}[x_i]$ to P_j, and continue to the next value of X. The total time for constructing $\mathsf{P}_1, \ldots, \mathsf{P}_k$ is clearly $O(n \log n)$. Finally, the number of sets k is at most $2\lceil n^{1-t} \rceil$ because if $|\mathsf{P}_i| < n^t$, then $|\mathsf{P}_{i+1}| \geq n^t$. The construction of the fractional-cascaded sequences to expedite searching in these sets is straightforward, and takes $O(n)$ time, as shown in [10].

The preprocessing cost of the uniform sets is only linear because computing $\pi(\mathsf{P}_j \cap (-\infty, a])$, for all $a \in \mathsf{P}_j$, takes $O(|\mathsf{P}_j|)$ time if P_j is uniform. The main part of the construction is the preprocessing of the non-uniform sets, which we describe in detail below.

Suppose $X_j = \langle x_1 < \ldots < x_s \rangle$ is the sequence of distinct values for the points of P_j. Let T be a complete binary tree whose leaves store x_j's in the sorted order and let $X_u \subseteq X_j$ be the subset of values stored at the leaves of the subtree rooted at a node $u \in T$. We will dynamically maintain a set $S \subseteq \mathsf{P}_j$ in T under insertion of points, and each node $u \in T$ will store $\pi_u = \pi(S_u)$ and $\mu_u = \mu(S_u)$, where $S_u = \{p_i \in S \mid v_i \in X_u\}$ is the set of points associated with values in X_u. More specifically, if u is a leaf, with $X_u = \{x\}$, then $\pi_u = \prod_{p_i \in S[x]}(1 - \alpha_i)$ and $\mu_u = x(1 - \pi(S_u))$. If u is an internal node whose left and right children are w and z, then we can see that (using the same argument as Lemma 2.1)

$$\pi_u = \pi_w \cdot \pi_z \quad \text{and} \quad \mu_u = \mu_z + \pi_z \mu_w. \tag{1}$$

When inserting a point p_i into S, we only need to update the nodes along the root-to-leaf path between the root of T and the leaf z storing v_i. Starting at the leaf, whose values π_z and μ_z are easily updated in $O(1)$ time, we update the remaining nodes of the path in the bottom-up order, using the equation (1). The

total time spent in updating T is $O(\log |\mathsf{P}_j|)$. With this structure in place, we can now describe how to finish the preprocessing of P_j.

Suppose the points of P_j are $p_1 < \ldots < p_r$, in left to right order. We need to compute $\pi(\cdot)$ and $\mu(\cdot)$ values for all of its $\binom{r}{2}$ intervals, in total time $O(r^2 \log r)$. We do this in batches where for a fixed $u \leq r$, we compute in $O(r \log r)$ time $\pi(\mathsf{P}_j \cap [p_u, p_v])$ and $\mu(\mathsf{P} \cap [p_u, p_v])$, for all v with $u \leq v \leq r$, as follows. We initialize $S = \emptyset$, and $\pi_v = 1$, $\mu_v = 0$ for all $v \in T$. Then we add $p_u, p_{u+1}, \ldots$ one by one. After $p_u, \ldots, p_v$ have been inserted, the values π_{root}, μ_{root} at the root of T are precisely $\pi(\mathsf{P}_j \cap [p_u, p_v])$ and $\mu(\mathsf{P}_j \cap [p_u, p_v])$. Because each insertion takes $O(\log r)$ time, the total time spent is $O(r \log r)$. We repeat this procedure for all $u \leq r$, which bounds the total preprocessing time for a non-uniform set P_j as $O(r^2 \log r) = O(n^{2t} \log n)$. We, therefore, have the following theorem.

Theorem 2.2 *Let P be a set of n uncertain points in $\mathbb{R}^1$. For any $t \in [0, 1]$, an index of size $O(n^{1+t})$ can be constructed in $O(n^{1+t} \log n)$ time so that $\mu(\mathsf{P} \cap I)$, for a query interval I, can be computed in $O(n^{1-t} + \log n)$ time.*

The 1-dimensional indexing algorithm can be extended to higher dimensions easily as follows. We begin by fixing a parameter t and partitioning the input into $k \leq 2\lceil n^{1-t} \rceil$ subsets $\mathsf{P}_1, \ldots, \mathsf{P}_k$ by value, as in one dimension. The main change needed is in lines 1-3 of Algorithm 1, where instead of "canonical intervals" we need to preprocess for canonical d-dimensional rectangles. In particular, for each non-uniform set P_j, there are now $|\mathsf{P}_j|^{2d} = O(n^{2dt})$ "combinatorially distinct" rectangles which we refer to as *canonical* rectangles. For each canonical rectangle ρ, we compute $\mu(\mathsf{P}_j \cap \rho)$ and $\pi(\mathsf{P}_j \cap \rho)$. By adapting the 1D preprocessing algorithm, these values can be computed in a total of $O(n^{2dt} \log n)$ time. The preprocessing of the uniform sets is the same as in 1-dimension, namely, sorting of points in each dimension.

Summing over all j, the total space and preprocessing time are $O(n^{(2d-1)t+1})$ and $O(n^{(2d-1)t+1} \log n)$ respectively. Given a query rectangle ρ, we "project" it onto its canonical rectangles $\rho_1, \ldots, \rho_k$, i.e., for each j a rectangle ρ_j such that, $\rho \cap \mathsf{P}_j = \rho_j \cap \mathsf{P}_j$. This requires finding the predecessor and successor within P_j of its extent in each dimension, which can be done in $O(\log n)$ time using fractional cascading. Since there are $k = n^{1-t}$ sets P_j, the total time to find all the canonical rectangles is $O(\log n + n^{1-t})$ using fractional cascading. The remaining part of Algorithm 1, namely, lines 4-9 then compute $\mu(\mathsf{P} \cap \rho)$ as before. We thus obtain the following:

Theorem 2.3 *Let P be a set of n uncertain points in $\mathbb{R}^d$. For any $t \in [0, 1]$, an index of size $O(n^{(2d-1)t+1})$ can be constructed in $O(n^{(2d-1)t+1} \log n)$ time so that $\mu(\mathsf{P} \cap \rho)$, for a query rectangle ρ, can be computed in $O(n^{1-t} + \log n)$ time.*

The index just described can be adapted easily to compute the most likely maximum value $\lambda(\mathsf{P} \cap I)$ with the same performance bounds. For each non-uniform set P_j and for each pair $p_u, p_v \in \mathsf{P}_j$ with $p_u < p_v$, we compute the values $x_{uv} = \lambda(\mathsf{P}_j \cap [p_u, p_v])$, $\gamma(x_{uv}, \mathsf{P}_j \cap [p_u, p_v])$, and $\pi(\mathsf{P}_j \cap [p_u, p_v])$. The query algorithm is a simple adaptation of Algorithm 1, leading to the following result:

Theorem 2.4 *Let P be a set of n uncertain points in $\mathbb{R}^d$. For any $t \in [0, 1]$, an index of size $O(n^{(2d-1)t+1})$ can be constructed in $O(n^{(2d-1)t+1} \log n)$ time so that $\lambda(\mathsf{P} \cap \rho)$, for a query rectangle ρ, can be computed in $O(n^{1-t} + \log n)$ time.*

Extensions. (i) Our index can be generalized to return the k values with the highest probabilities of being the range-max for a query rectangle ρ, as follows. For each combinatorially distinct rectangle, we store the k most likely maximum values instead of the single most likely maximum leading to an index of size $O(kn^{(2d-1)t+1})$. Given a query rectangle ρ, we search among the k most-likely values stored at the relevant canonical rectangle in each subset P_i and find the k most likely values in $O(n^{1-t} + k \log n)$ time.

(ii) The algorithm can be extended to the case in which each point p_i is associated with a discrete distribution over its values of constant support f, i.e., $\{v_{i1}, \ldots, v_{if}\} \subset \mathbb{R}^+$ with probabilities $\{\alpha_{i1}, \ldots, \alpha_{if}\} \subset [0, 1]$, where the point assumes value v_{ij} with probability α_{ij}. An instance of the EM (MLM) problem with n points of value uncertainty can be mapped to an instance of EM (MLM) problem with nf independent points of existential uncertainty; we omit the details from this abstract.

3. HARDNESS OF MLM

The range-max query problem for uncertain data seems significantly harder than its deterministic counterpart. Indeed, the latter can be solved easily in $O(\log^{d-1} n)$ query time and $O(n \log^{d-1} n)$ space using orthogonal range trees with fractional cascading, for $d \geq 2$ [2]. This stands in sharp contrast to what we are able to achieve for the uncertain data model; to wit, if the space requirement of our data structure is nearly linear, our query time goes up to near linear as well. In this section, we offer evidence for the hardness of uncertain range-max problem through a reduction from the *set-intersection* problem, which suggests that indexes with near-linear space and poly-logarithmic query time are unlikely to exist even for $d = 2$. Our lower bound is in the real-RAM model, mentioned above.

The set-intersection problem is defined as follows. Given a family of sets $S_1, S_2, \ldots, S_m$, with real-valued items, preprocess them so that the intersection queries of the following form can be answered efficiently: given indices i, j, do the sets S_i and S_j have a non-empty intersection? It is widely believed that any index that can answer such queries in worst-case time $O(t)$ must

use at least $\Omega((n/t)^2)$ space, ignoring polylog factors [15, 23, 24], where $n = \sum |S_i|$ is the total size of all the sets. We show in the following that any index for solving the MLM problem in $\mathbb{R}^d$ for $d \geq 2$ can also solve the set-intersection problem, and is therefore subject to the same query-space tradeoff.

Reduction. We transform an instance of the set-intersection problem of size n into an instance of the MLM problem with $2n$ points in $\mathbb{R}^2$. Specifically, let $S_1, \ldots, S_m$ be the input to the set-intersection problem, where $\sum_{i=1}^{m} |S_i| = n$. Let $n_0 = 0$ and $n_i = n_{i-1} + |S_i|$ for $i = 1, \ldots, m$. We use s_{ik} to denote the (value of) the kth item of S_i, for $1 \leq k \leq |S_i|$. We create a 2-dimensional instance of the MLM problem corresponding to these sets as follows. All the points in the MLM lie on two parallel lines, $L : y = x + n$ and $L' : y = x - n$. For each member of the set S_i, we create two points, one on each of these lines. Specifically, the points corresponding to the kth item $s_{ik} \in S_i$ are: $(-(k + n_{i-1}), -(k + n_{i-1}) + n))$ on L, and $((k + n_{i-1}), (k + n_{i-1}) - n))$ on L'. The existence probability of these two points is $\alpha = 1 - 2^{-\frac{1}{n+1}}$ and their value is s_{ik}. Let P_i and P'_i denote the set of points corresponding to S_i that lie on L and L', respectively. Finally, let $\mathsf{P} = \bigcup_i (\mathsf{P}_i \cup \mathsf{P}'_i)$ denote the set of all the $2n$ points that form the input to the MLM problem. Note that all points of P have the same existence probability α, and for any pair i, j, all points in $\mathsf{P}_i \cup \mathsf{P}'_j$ have distinct values if and only if $S_i \cap S_j = \emptyset$.

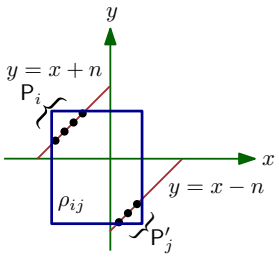

Figure 2. Illustration for the lower bound construction.

Clearly the construction takes $O(n)$ time, and one can easily verify that (1) all points on L lie in the northwest quadrant while those on L' lie in the southeast quadrant, and (2) the points corresponding to S_i are placed in consecutive order on both L and L', and they lie between the points corresponding to S_{i-1} and S_{i+1}.

The placement of the points in opposite quadrants ensures the following geometric property: given any two indices i, j, there is a rectangular range ρ_{ij} that includes only P_i and P'_j, namely, $\rho_{ij} \cap \mathsf{P} = \mathsf{P}_i \cup \mathsf{P}'_j$. See Figure 2.

Given a set-intersection query: "Does S_i intersect S_j?" we compute the MLM for the query range ρ_{ij}. If the answer is value v, then we check if v belongs to both S_i and S_j. If it does, then we return "Yes," meaning that the sets intersect; otherwise, we answer "No."

Thus, we answer a set-intersection query using one MLM range query and two set membership queries to decide if $v \in S_i \cap S_j$. The membership queries are easily performed in $O(\log n)$ time, using any standard balanced search tree, and so the set-intersection query time is dominated by the MLM query time, as long as the latter is $\Omega(\log n)$. The correctness of the reduction follows from the following lemma.

Lemma 3.1 *In the preceding reduction, $S_i \cap S_j \neq \emptyset$ if and only if $\lambda(\mathsf{P} \cap \rho_{ij})$ belongs to $S_i \cap S_j$.*

PROOF. If $\lambda(\mathsf{P} \cap \rho_{ij})$ belongs to $S_i \cap S_j$, then $S_i \cap S_j \neq \emptyset$, and the claim holds trivially. Conversely, assume $S_i \cap S_j \neq \emptyset$. Let $\mathsf{P}_{ij} = \mathsf{P} \cap \rho_{ij}$, $v = \lambda(\mathsf{P}_{ij})$, and r the number of points in P_{ij} with values larger than v. If v is associated with only one point of P_{ij}, then $\gamma(v, \mathsf{P}_{ij}) = \alpha(1 - \alpha)^r \leq \alpha$.

On the other hand if v is associated with two points of P_{ij} (note that each value of P_{ij} is associated with at most two points), then

$$\gamma(v, \mathsf{P}_{ij}) = (2\alpha - \alpha^2)(1 - \alpha)^r \geq (2\alpha - \alpha^2)(1 - \alpha)^n > \alpha$$

because $\alpha = 1 - 2^{-\frac{1}{n+1}}$. Since $S_i \cap S_j \neq \emptyset$, there are two points in P_{ij} with the same value, corresponding to an item of $S_i \cap S_j$. Therefore v belongs to $S_i \cap S_j$ in this case, as claimed. □

We have established the following result.

Theorem 3.2 *If there is an index to solve the MLM problem for n points in $\mathbb{R}^2$ with $S(n)$ space and $Q(n)$ query time, for $Q(n) \geq \Omega(\log n)$, then the set-intersection problem can be solved in $Q(2n)$ time with space $S(2n)$.*

The set intersection problem is conjectured to require $\Omega((n/t)^2)$ space, ignoring polylog factors, as long as the query time is $t = \Omega(\log n)$. Subject to this conjecture, Theorem 3.2 implies that any data structure that uses near-linear space cannot hope to achieve polylog query time. More formally, the theorem suggests that in the worst-case the product (query time)^2 × (index size) is $\Omega(n^2)$, ignoring polylog factors, for $d \geq 2$.

4. RANDOM PEMUTATION MODEL

We describe an index of $O(n \,\mathrm{polylog}(n))$ size that can answer an MLM query in $O(\mathrm{polylog}(n))$ expected time if the values of points are distinct and they are assigned using the so-called *random permutation model*. More precisely, we assume that the existence probabilities α_i's of points are arbitrary but their *values* v_i's are assigned randomly as follows: Let $x_1 \geq x_2 \geq \ldots \geq x_n$ be an arbitrary set of real values. We choose a random permutation σ of $[1 : n]$ and set $v_i = x_{\sigma(i)}$.

We map each point $p_i \in \mathsf{P}$ to a two-dimensional point $\hat{p}_i = (\alpha_i, v_i) \in \mathbb{R}^2$. Let $\hat{\mathsf{P}} = \{\hat{p}_i \mid p_i \in \mathsf{P}\}$. A point $a = (a_1, a_2)$ *dominates* another point $b = (b_1, b_2)$ if $a_1 > a_2$ and $b_1 \geq b_2$ or $a_1 \geq a_2$ and $b_1 > b_2$. The *skyline* of a point set $S \subseteq \mathbb{R}^2$ is the subset of points in S that

are not dominated by another point of S. The following simple lemma is the basis of our index.[2]

Lemma 4.1 *Let $\mathsf{R} \subseteq \mathsf{P}$ such that the values of all points in R are distinct. If $v_k = \lambda(\mathsf{R})$, then $\hat{p}_k$ belongs to the skyline of $\hat{\mathsf{R}} = \{\hat{p}_i \mid p_i \in \mathsf{R}\}$.*

PROOF. If $\hat{p}_k$ is not on the skyline of $\hat{\mathsf{R}}$, then it is dominated by some other point $\hat{p}_j \in \hat{\mathsf{R}} \backslash \{\hat{p}_k\}$. Thus, we have $\alpha_j \geq \alpha_k$ and $v_j \geq v_k$. Since the values are distinct, the second inequality must be strict, namely, $v_j > v_k$. It is easy to check that

$$\beta(v_k, \mathsf{R}) \leq (1 - \alpha_j)\beta(v_j, \mathsf{R}).$$

Therefore, we have the following inequality:

$$\begin{aligned}\gamma(v_k, \mathsf{R}) &= \alpha(v_k, \mathsf{R})\beta(v_k, \mathsf{R}) \leq \alpha_k(1 - \alpha_j)\beta(v_j, \mathsf{R}) \\ &< \alpha_j\beta(v_j, \mathsf{R}) = \gamma(v_j, \mathsf{R})\end{aligned}$$

which contradicts the assumption that

$$\gamma(v_k, \mathsf{R}) = \operatorname*{argmax}_{x \in X} \gamma(x, \mathsf{R}).$$

This completes the proof. □

For a rectangle ρ, let $\mathsf{P}_\rho = \mathsf{P} \cap \rho$, $\hat{\mathsf{P}}_\rho = \{\hat{p} \mid p \in \mathsf{P}_\rho\}$, and $S_\rho \subseteq \mathsf{P}_\rho$ the set of points p_i such that $\hat{p}_i$ belongs to the skyline of $\hat{\mathsf{P}}_\rho$. By Lemma 4.1,

$$\lambda(\mathsf{P}_\rho) = \arg\max_{p_i \in S_\rho} \gamma(v_i, \mathsf{P}_\rho).$$

If the values of P are assigned using a random permutation model, then we note that the same holds for an arbitrary subset $\mathsf{R} \subseteq \mathsf{P}$ as well. A classical result in stochastic geometry implies that the expected size of the skyline of $\hat{\mathsf{P}}_\rho$ is $O(\log n)$, see e.g. [20, 7]. Using these observations, we compute $\lambda(\mathsf{P}_\rho)$ in two stages: First, we compute S_ρ, and then compute $\gamma(v_i, \mathsf{P}_\rho)$ for every $p_i \in S_\rho$.

Rahul and Janardan [24] have presented an index for storing P and $\hat{\mathsf{P}}$ of size $O(n \log^{d+1} n)$, that can be constructed in time $O(n \log^{d+1} n)$ so that for any query rectangle ρ, S_ρ can be computed in time $O(|S_\rho| \log^{d+2} n))$. For $d = 1$, the size, the preprocessing time, and the query time are $O(n)$, $O(n \log^2 n)$ time, and $O(|S_\rho| + \log n)$, respectively.

To compute $\gamma(v_i, \mathsf{P}_\rho)$, for every $p_i \in S_\rho$, we map each point $p_i \in \mathsf{P}$ to $\bar{p}_i := (p_i, v_i) \in \mathbb{R}^{d+1}$ and set its weight $w(\bar{p}_i) = 1 - \alpha_i$. Set $\bar{\mathsf{P}} = \{\bar{p}_i \mid 1 \leq i \leq n\}$. We build, in $O(n \log^d n)$ time, a $(d+1)$-dimensional range tree of size $O(n \log^d n)$ so that for any $(d+1)$-dimensional rectangle $\bar{\rho}$, $w(\bar{\rho}) = \prod_{\bar{p} \in \bar{\mathsf{P}} \cap \bar{\rho}} w(\bar{p})$ can be computed in $O(\log^d n)$ time [2]. For $p_i \in S_\rho$, let $\bar{\rho}_i = \rho \times (v_i, \infty)$. Then $\gamma(v_i, \mathsf{P}_\rho) = \alpha_i w(\bar{\rho}_i)$ can be computed in $O(\log^d n)$ time using this index.

Putting everything together, we obtain the following result:

[2]The lemma does not assume the values to be assigned using a random permutation. It holds for arbitrarily assigned values, as long as they are distinct.

Theorem 4.2 *Let P be a set of n uncertain points in $\mathbb{R}^d$, such that the values of P are all distinct and chosen according to a random permutation model. An index of size $O(n \log^{d+1} n)$ on P can be built in $O(n \log^{d+1} n)$ time so that for a query rectangle ρ, $\lambda(\mathsf{P} \cap \rho)$ can be computed in expected time $O(\log^{d+3} n)$. For $d = 1$, the size and expected query time can be improved to $O(n)$ and $O(\log^2 n)$, respectively.*

5. APPROXIMATION ALGORITHMS

In this section we describe approximation algorithms for EM and MLM problems. We begin by discussing two simple algorithms. The first one can be used for both EM and MLM problems but the running time depends on the probabilities of the points. The second one works only for MLM and gives an additive approximation. In the end, we propose a constant-factor approximation algorithm for the EM problem, whose running time is independent of the input probabilities.

Suppose the existence probability of each point in P is at least α. Set $t = \left\lceil \frac{1}{\alpha} \ln \frac{n}{\varepsilon\alpha} \right\rceil$ for any $\varepsilon > 0$. A simple calculation shows that for any rectangle ρ and for any value $x \in X$ that is not among the t largest ones in $\mathsf{P}_\rho = \mathsf{P} \cap \rho$, $\gamma(x, \mathsf{P}_\rho) \leq \frac{\varepsilon\alpha}{n}$, i.e., it is negligible and such a point can be ignored. Using a standard index for reporting top t values inside ρ [2], we can construct an index of size $O(n \log^{d-1} n)$ in time $O(n \log^{d-1} n)$ that returns, in $O(\log^{d-1} n + t \log n)$ time, values $\eta_\rho \in \mathbb{R}^+$ and $x_\rho \in X$ such that $(1 - \varepsilon)\mu(\mathsf{P}_\rho) \leq \eta_\rho \leq \mu(\mathsf{P}_\rho)$ and $\gamma(x_\rho, \mathsf{P}_\rho) \geq (1 - \varepsilon) \max_{x \in X} \gamma(x, \mathsf{P}_\rho)$.

If there are input points with small existence probabilities, for example $\alpha = \frac{1}{\sqrt{n}}$, then the previous scheme is not efficient. In these cases a straightforward sampling-based Monte Carlo approach can estimate $\gamma(x, \mathsf{P}_\rho)$ with additive error ε, for any $x \in X$ and rectangle ρ, with high probability. We instantiate P, $s = O(\frac{1}{\varepsilon} \log n)$ times. Each time the point p_j is chosen with probability α_j. Let S_i be the i-th instance of P. For each $i \leq s$, we build a range-MAX index, like a range-tree, for the set S_i. Given a query rectangle ρ, we find the maximum value in $\rho \cap S_i$ for each i, and return the value that was the maximum in most of the indexes. This procedure returns a value x_ρ such that $\gamma(x_\rho, \mathsf{P}_\rho) \geq \max_{x \in X} \gamma(x, \mathsf{P}_\rho) - \varepsilon$ with high probability, in time $O(\frac{1}{\varepsilon} \log^d n)$, using $O(\frac{1}{\varepsilon} n \log^d n)$ space.

This simple sampling-based approach, however, does not work for estimating $\mu(\mathsf{P}_\rho)$, for a query rectangle, because a point in P_ρ with very small existence probability may have a very large value. We describe a novel $O(n \operatorname{polylog}(n))$-size index that estimates $\mu(\mathsf{P}_\rho)$ within relative error $1/2$ in $O(\operatorname{polylog}(n))$ time. The index is based on the following lemma.

Lemma 5.1 *Let $Y_1, \ldots, Y_m$ be Bernoulli random variables such that Y_i assumes value v_i with probability α_i and 0 otherwise. Then,*

$$\mathbf{E}\left[\max_j Y_j\right] \geq w \geq \frac{1}{2}\mathbf{E}\left[\max_j Y_j\right],$$

where w is a solution of the equation

$$w = \sum_j \mathbf{E}\left[\max\{Y_j - w, 0\}\right].$$

The second part of this inequality is the well-known Prophet inequality [25]. The first part of the inequality can be shown by induction on i; we omit the proof from this abstract.[3] We now discuss how to use this lemma to devise an index for the range-max queries.

For a point $p_j \in \mathsf{P}$, let Y_j be a random variable that has value v_j with probability α_j and 0 otherwise. For a subset $\mathsf{R} \subseteq \mathsf{P}$ and a real-valued parameter $t > 0$, we define

$$F_{\mathsf{R}}(t) = \sum_{p_j \in \mathsf{R}} \mathbf{E}\left[\max\{Y_j - t, 0\}\right]. \tag{2}$$

Let $X = \langle x_1, \ldots, x_m \rangle$ be the sequence of distinct values of points of P in decreasing order. The proof of the following lemma is straightforward.

Lemma 5.2 *For any subset $\mathsf{R} \subseteq \mathsf{P}$,*

(i) $F_{\mathsf{R}}(t)$ is a monotonically decreasing, piecewise-linear continuous function.

(ii) For any $1 \le i \le m$, $F_{\mathsf{R}}(t)$ is a linear function in the interval $[x_i, x_{i-1}]$ with the following form:

$$F_{\mathsf{R},i}(t) = \sum_{p_j \in \mathsf{R}: v_j \ge x_{i-1}} \alpha_j (v_j - t). \tag{3}$$

The following corollary is an easy consequence of Lemma 5.2 (i).

Corollary 5.3 *There is a unique value t, denoted by $\tau(\mathsf{R})$, for which $F_{\mathsf{R}}(t) = t$.*

Given a subset $\mathsf{R} \subseteq \mathsf{P}$, $\tau(\mathsf{R})$ can be computed in two stages. The first stage computes the value $i := i(\mathsf{R})$ such that $\tau(\mathsf{R}) \in [x_i, x_{i-1}]$ by performing a binary search on X. The second stage computes $\tau(\mathsf{R})$ by solving the linear equation $F_{\mathsf{R},i}(t) - t = 0$ (cf. 3).

Indexing scheme. Define $\bar{\mathsf{P}} = \{\bar{p}_i \mid 1 \le i \le n\}$, where $\bar{p}_i = (p_i, v_i) \in \mathbb{R}^{d+1}$, for $p_i \in \mathsf{P}$. We build an index on $\bar{\mathsf{P}}$ so that for any query $(d+1)$-dimensional rectangle $\bar{\rho}$, quantities $b_{\bar{\rho}} = \sum_{\bar{p}_i \in \bar{\rho} \cap \bar{\mathsf{P}}} \alpha_i v_i$ and $a_{\bar{\rho}} = \sum_{\bar{p}_i \in \bar{\rho} \cap \bar{\mathsf{P}}} \alpha_i$ can be computed quickly, using an instance of a $(d+1)$-dimensional range-SUM query. In particular, an index of size $O(n \log^d n)$ can be built in time $O(n \log^d n)$ so that $a_{\bar{\rho}}$, $b_{\bar{\rho}}$ can be computed in $O(\log^d n)$ time [2].

Query procedure. Let ρ be a query rectangle in $\mathbb{R}^d$, and we wish to estimate $\mu(\mathsf{P}_\rho)$. We perform a binary search on X to compute $i := i(\mathsf{P}_\rho)$ such that $\tau(\mathsf{P}_\rho) \in [x_i, x_{i-1}]$. Each step of the binary search chooses a value $x_r \in X$ and queries the index with the rectangle $\bar{\rho}_r = \rho \times [x_r, \infty)$, which returns

$$a_{\bar{\rho}_r} = \sum_{\bar{p}_i \in \bar{\rho}_r \cap \bar{\mathsf{P}}} \alpha_i, \quad b_{\bar{\rho}_r} = \sum_{p_i \in \bar{\rho}_r \cap \bar{\mathsf{P}}} \alpha_i v_i. \tag{4}$$

By Lemma 5.2

$$F_{\mathsf{P}_\rho}(x_r) = b_{\bar{\rho}_r} - a_{\bar{\rho}_r} x_r. \tag{5}$$

Therefore, by comparing $F_{\mathsf{P}_\rho}(x_r)$ with x_r, we can determine in $O(1)$ time whether $\tau(\mathsf{P}_\rho) = x_r$, $\tau(\mathsf{P}_\rho) > x_r$, or $\tau(\mathsf{P}_\rho) < x_r$. In the first case, we have the value of $\tau(\mathsf{P}_\rho)$ and return $F_{\mathsf{P}_\rho}(x_r)$ as an estimate of $\mu(\mathsf{P}_\rho)$. Otherwise, we proceed with the binary search.

After having computed the value $i(\mathsf{P}_\rho)$, we can compute $\tau(\mathsf{P}_\rho)$ in another $O(1)$ time. Putting everything together, we obtain the following:

Theorem 5.4 *Let P be a set of n uncertain points in $\mathbb{R}^d$. An index of size $O(n \log^d n)$ can be built on P in time $O(n \log^d n)$ that for a query rectangle ρ returns in $O(\log^{d+1} n)$ time a value η_ρ such that $\frac{1}{2}\mu(\mathsf{P} \cap \rho) \le \eta_\rho \le \mu(\mathsf{P} \cap \rho)$.*

6. LOCATION UNCERTAINTY MODEL

In the interest of exposition and to highlight the conceptual framework, we have described our range-max index structures for a simple model of data uncertainty. The methodology, however, extends naturally to more general uncertainty models. We already discussed one such extension—uncertainty in values—at the end of Section 2. In this section, we discuss *location uncertainty*, where the position of each point has a probability distribution.

In the location-uncertainty model, the location of a point P is represented as a probability distribution. We assume P is specified by a point set $\{p_1, \ldots, p_f\}$ with associated probabilities $\{\alpha_1, \ldots, \alpha_f\}$, i.e., $\alpha_j = \mathbf{Pr}\big[P \text{ is at location } p_j\big]$, $\sum_{j=1}^{f} \alpha_j \le 1$, and P does not appear with probability $1 - \sum_{j=1}^{f} \alpha_j$. We refer to f as the *support size* of P. Let $\mathsf{P} = \{P_1, \ldots, P_n\}$ be a set of n points in $\mathbb{R}^d$ in the location uncertainty model. Without loss of generality, we assume that the support size of each point is f, i.e., $P_i = \{p_{i1}, \ldots, p_{if}\}$ and $\alpha_{ij} = \mathbf{Pr}\big[P_i \text{ is at } p_{ij}\big]$.

Each P_i is associated with a scalar $v_i > 0$. Note that the model described in Section 1 is a special case of this model with $f = 1$. With a slight abuse of notation we will use P_i to define an uncertain point as well as the set of its possible locations. We wish to compute the expected or most-likely value of the maximum in a query rectangle ρ.

The algorithms described in Sections 2, 4, and 5 can be extended to the location uncertainty model. Since the location uncertainty model is more general, the lower bound described in Section 3 holds for this model but we also prove a similar lower bound for the EM problem in this model. Due to lack of space we only describe how to extend the exact and approximation algorithms, and we briefly sketch the lower-bound construction.

[3] We thank Kamesh Munagala for pointing out the usage of Prophet inequality.

Exact algorithm. For simplicity we assume the values of points in P are distinct; the case of multiple points having the same value can be handled by adapting the definitions and Algorithm 1 in Section 2. We show that the expected and the most-likely value of maximum inside a rectangle can be defined analogously to Section 2.

For a rectangle ρ, let $P_i(\rho)$ denote the restriction of P_i inside ρ. That is, $P_i(\rho)$ is an uncertain point with possible locations $P_i \cap \rho$ and for $p_{ij} \in P_i \cap \rho$, $\mathbf{Pr}\big[P_i(\rho) \text{ is at } p_{ij}\big] = \alpha_{ij}$. We define $\alpha_i(\rho) = \sum_{p_{ij} \in P_i \cap \rho} \alpha_{ij}$ to be the probability of P_i appearing inside ρ, and $\pi_i(\rho) = 1 - \alpha_i(\rho)$ to be the probability of P_i not appearing inside ρ. For a subset $\mathsf{Q} \subseteq \mathsf{P}$ and for a rectangle ρ let $\gamma_i(\mathsf{Q}, \rho)$ be the probability of v_i, with $P_i \in \mathsf{Q}$, being the maximum value among the points of Q in ρ, then [4]

$$\gamma_i(\mathsf{Q}, \rho) = \alpha_i(\rho) \prod_{P_j \in \mathsf{Q}: v_j > v_i} \pi_j(\rho),$$

$$\mu(\mathsf{Q}, \rho) = \sum_{i=1}^{n} v_i \gamma_i(\mathsf{Q}, \rho),$$

$$\lambda(\mathsf{Q}, \rho) = \arg\max_{1 \le i \le n} \gamma_i(\mathsf{Q}, \rho).$$

With these definitions γ_i, μ, and λ in place, the overall structure of the algorithm is the same as before. We partition P into $O(n^{1-t})$ subsets $\mathsf{P}_1, \ldots, \mathsf{P}_k$, each of size at most $2n^t$, so that all points in P_i have higher values than those in P_{i+1}. For each P_i, there is a set $S_i = \bigcup_{P_j \in \mathsf{P}_i} P_j$ of $O(n^t f)$ possible locations. Therefore, there are $O((n^t f)^{2d})$ "combinatorially distinct" rectangles with respect to P_i, each defined by a subset of $2d$ locations in S_i, in the sense that $\gamma(\mathsf{P}_i, \rho)$, $\mu(\mathsf{P}_i, \rho)$, and $\lambda(\mathsf{P}_i, \rho)$, for any rectangle ρ, are the same as for one of these rectangles. We store $\mu(\mathsf{P}_i, \rho)$ and $\pi(\mathsf{P}_i, \rho) = \prod_{P_j \in \mathsf{P}_i} \pi_j(\rho)$ for every combinatorially distinct rectangle. The total size of the index is $O(n^{(2d-1)t+1} f^{2d})$, and it can be constructed in time $O(n^{(2d-1)t+1} f^{2d} \log n)$. Algorithm 1 can be adapted in a straightforward manner to compute $\mu(\mathsf{P}, \rho)$ and $\lambda(\mathsf{P}, \rho)$ in time $O(n^{1-t} + \log(nf))$. We thus obtain the following:

Theorem 6.1 *Let P be a set of n locationally uncertain points in $\mathbb{R}^d$, each of support size f. For any $t \in [0, 1]$, an index of size $O(n^{(2d-1)t+1} f^{2d})$ can be constructed in $O(n^{(2d-1)t+1} f^{2d} \log n)$ time so that the expected or most-likely maximum value inside a query rectangle can be computed in $O(n^{1-t} + \log(nf))$ time.*

Approximation algorithm. We now show that the approximation algorithm described in Section 5 for the EM problem can be extended to the location uncertainty model.

For a rectangle ρ and for a point $P_i \in \mathsf{P}$, we define a random variable $Y_{i,\rho}$ whose value is v_i if P_i lies inside ρ and 0 otherwise. Then

$$\mathbf{Pr}\big[Y_{i,\rho} = v_i\big] = \alpha_i(\rho) = \sum_{p_{ij} \in P_i(\rho)} \alpha_{ij}.$$

and $\mu(\mathsf{P}, \rho) = \mathbf{E}\left[\max Y_{j,\rho}\right]$. Lemma 5.1 still holds for the Bernoulli random variables $Y_{i,\rho}$. We also define the function

$$F_\rho(t) = \sum_{j=1}^{n} \mathbf{E}\left[\max\{Y_{j,\rho} - t, 0\}\right].$$

Let $v_1, \ldots, v_n$ be the values of points in decreasing order, then analogously to Lemma 5.2, $F_\rho(t)$ has the following form in $[v_i, v_{i-1}]$:

$$F_{\rho,i}(t) = \sum_{v_j \ge v_{i-1}} \alpha_j(\rho)(v_j - t) := b_{\rho,i} - a_{\rho,i} t,$$

where

$$a_{\rho,i} = \sum_{v_j \ge v_{i-1}} \alpha_j(\rho), \quad b_{\rho,i} = \sum_{v_j \ge v_{i-1}} \alpha_j(\rho) v_j.$$

As in Section 5, we construct an index to compute $a_{\rho,i}$, $b_{\rho,i}$ quickly for a given rectangle ρ and an integer $i \in [2, n]$. More precisely, for $1 \le i \le n$ and $1 \le j \le f$, let $\bar{p}_{ij} = (p_{ij}, v_i) \in \mathbb{R}^{d+1}$. We assign a pair of weights $w_a(\bar{p}_{ij}) = \alpha_{ij}$ and $w_b(\bar{p}_{ij}) = \alpha_{ij} v_j$ for each $\bar{p}_{ij}$. Set $\bar{S} = \{\bar{p}_{ij} \mid 1 \le i \le n, 1 \le j \le f\}$. For a rectangle ρ in $\mathbb{R}^d$ and for an integer $i \in [2, n]$, let $\bar{\rho}_i = \rho \times [v_{i-1}, \infty)$. Then

$$a_{\rho,i} = \sum_{\bar{p}_{jk} \in \bar{S} \cap \bar{\rho}_i} w_a(\bar{p}_{jk}), \quad b_{\rho,i} = \sum_{\bar{p}_{jk} \in \bar{S} \cap \bar{\rho}_i} w_b(\bar{p}_{jk}).$$

By building an index on $\bar{S}$ for range-SUM queries, $a_{\rho,i}$, $b_{\rho,i}$ can be computed in $O(\log^d n)$ time. The size of the index is $O(nf \log^d(nf))$, and it can be constructed in the same time [2]. The query procedure is the same as in Section 5. We thus obtain the following:

Theorem 6.2 *Let P be a set of n locationally uncertain points in $\mathbb{R}^d$, each with support size f. An index of size $O(nf \log^d(nf))$ can be constructed in time $O(nf \log^d(nf))$ that for a query rectangle ρ returns in $O(\log^{d+1}(nf))$ time a value η_ρ, such that $\frac{1}{2}\mu(\mathsf{P}, \rho) \le \eta_\rho \le \mu(\mathsf{P}, \rho)$.*

Lower bound. We sketch the reduction of the set-intersection problem to the EM problem[5]. We may assume all the integers in the sets $S_1, \ldots, S_m$ are divisible by 3. If not, we can work with the instance $S_i' = \{3x \mid x \in S_i\}$, without a change to the answer of any set-intersection query. We use almost the same construction as in the proof of Theorem 3.2, with four important differences.

[4] In the location uncertainty model, P is a family of sets of points, so the notation here is slightly different from earlier sections.

[5] In the lower bound, we use sets with *integer-valued* items, as opposed to real-valued items. However, the set-intersection problem over reals is easily reduced to an integer-valued instance, and so the former is just as hard [15, 23, 24].

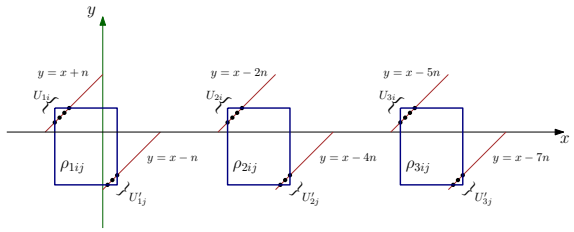

Figure 3. Illustration - lower bound for EM in location uncertainty model.

(i) The entire configuration is shifted parallel to itself by $3n$ and $6n$ to get three non-interfering configurations;

(ii) the associated values of the points above the x-axis come from the sets as before, but for the points below the x-axis they are increased by 1 for the second instance, by 2 for the third instance and are unchanged in the first instance;

(iii) points with the same value are all considered as belonging to one uncertain point in the location model, and,

(iv) all points are assigned probability α which can be chosen small enough so that each uncertain point is deficient, i.e., the sum of probabilities of locations is less than 1, see Figure 3 for illustration.

To answer a query, "Does S_i intersect S_j?" where $i \neq j$, we consider the rectangles $\rho_{1ij}, \rho_{2ij}, \rho_{3ij}$ as shown in Figure 3, and we query the EM value for each of them - denote these by Ψ_1, Ψ_2, Ψ_3. We can show that:

Lemma 6.3 *$\Psi_2 = (\Psi_1 + \Psi_3)/2$ if and only if the sets S_i, S_j do not intersect.*

Intuitively, we can see the above as follows. The EM value $\mu(\mathsf{P}, \rho_{ij})$ is the sum $\sum_{k=1}^{n} v_k \gamma_k(\mathsf{P}, \rho_{ij})$, and if the queried sets S_i, S_j are disjoint, all the values occurring inside all the rectangles are distinct and one can easily show that the sum for each of the Ψ_k is formed by two distinct components - one coming from contributions from values for points corresponding to S_i and the other by those from S_j, and thus $\Psi_2 - \Psi_1 = \Psi_3 - \Psi_2$, because of the equal shifts in values. On the other hand, if the sets intersect, some points inside the first rectangle belong to the same uncertain point - but this does not happen for the second and third rectangles because the numbers involved are distinct modulo 3. In this case, Ψ_2, Ψ_3 behave similar to the case when S_i, S_j are disjoint but Ψ_1 increases. Thus we have, $\Psi_2 - \Psi_1 < \Psi_3 - \Psi_2$. Finally, we can show the following result where n, m come from the construction, as there are $O(n)$ uncertain points now, with support size bounded by $O(m)$.

Theorem 6.4 *Suppose there exists an index with size $S(n, f)$ and query time $Q(n, f)$ for solving the EM problem for n points in $\mathbb{R}^2$ under the location uncertainty model, where each point has a distribution over at most f possible locations. Then an instance of the set-intersection problem, for sets $S_1, \ldots, S_m$, with $\sum_{i=1}^{m} |S_i| = n$, can be solved in $O(Q(n, m))$ query time using space $O(S(n, m))$.*

7. CONCLUSION

In this paper, we studied the range-max query problem for uncertain data, and designed efficient index structures for answering these queries both exactly and approximately under two models of data uncertainty (existential and locational). Our exact algorithms are the first to achieve sublinear query time using sub-quadratic space, while our approximation algorithms utilize novel tools such as the Prophet inequality. We also prove a hardness result for the MLM, which shows that range-max queries over uncertain data are provably more difficult than their deterministic counterparts. Our work suggests several avenues of future research, including the following:

(i) Our query-space bounds represent an important advance but are unlikely to be optimal, so improving them is an obvious goal. In particular, is sublinear query time possible for the EM and MLM problems using only a linear size index?

(ii) Our lower bound for the MLM problem is conditioned on the (widely) conjectured hardness of the set intersection problem. Is an *unconditional* lower bound possible? Can the lower bound be shown for dimension $d = 1$?

(iii) Our Prophet Inequality based approximation works only for the EM problem. Can a similar scheme be designed for the MLM problem?

(iv) Extend our results to other range queries, such as range-SUM. Can we efficiently approximate the *distribution* of the range-SUM statistic?

Acknowledgments. We thank Kamesh Munagala for pointing out the prophet inequality, and Ke Yi and Wuzhou Zhang for helpful discussions.

8. REFERENCES

[1] A. Abdullah, S. Daruki, and J. M. Phillips. Range counting coresets for uncertain data. In *Proc. 29th Annu. Sympos. Comput. Geom.*, pages 223–232, 2013.

[2] P. K. Agarwal. Range searching. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Edition.*, pages 809–837. CRC, 2004.

[3] P. K. Agarwal, B. Aronov, S. Har-Peled, J. M. Phillips, K. Yi, and W. Zhang. Nearest neighbor searching under uncertainty ii. In *Proc. 32nd Annu. sympos. Princ. Database Sys.*, pages 115–126, 2013.

[4] P. K. Agarwal, S.-W. Cheng, and K. Yi. Range searching on uncertain data. *ACM Trans. Algorithms*, 8(4):43, 2012.

[5] P. K. Agarwal, A. Efrat, S. Sankararaman, and W. Zhang. Nearest neighbor searching under uncertainty. In *Proc. 31st Annu. ACM sympos. Princ. Database Sys.*, pages 225–236, 2012.

[6] C. C. Aggarwal. *Managing and Mining Uncertain Data: 3, A.*, volume 35. Springer Science & Business Media, 2010.

[7] Z. Bai, L. Devroye, H. Hwang, and T. Tsai. Maxima in hypercubes. *Random Structures & Algorithms*, 27(3):290–309, 2005.

[8] T. Bernecker, T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, and A. Zuefle. A novel probabilistic pruning approach to speed up similarity queries in uncertain databases. In *Proc. 27th Annu. IEEE Int. Conf. Data Eng.*, pages 339–350, 2011.

[9] G. Beskales, M. A. Soliman, and I. F. Ilyas. Efficient search for the top-k probable nearest neighbors in uncertain databases. *Proc. 34th Very Large Data Bases*, 1(1):326–339, 2008.

[10] B. Chazelle and L. J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(1-4):133–162, 1986.

[11] J. Chen and L. Feng. Efficient pruning algorithm for top-k ranking on dataset with value uncertainty. In *Proc. 22nd ACM Conf. Inf. Knowl. Manag.*, pages 2231–2236, 2013.

[12] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, and J. S. Vitter. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *Proc. 30th Very Large Data Bases*, pages 876–887, 2004.

[13] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *Proc. 25th Annu. IEEE Int. Conf. Data Eng.*, pages 305–316, 2009.

[14] N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: diamonds in the dirt. *Comm. ACM*, 52(7):86–94, 2009.

[15] P. Davoodi, M. Smid, and F. van Walderveen. Two-dimensional range diameter queries. In *Proc. 10th Latin American Theo. Inform. Sympos.*, pages 219–230, 2012.

[16] T. Ge, S. Zdonik, and S. Madden. Top-k queries on uncertain data: on score distribution and typical answers. In *Proc. SIGMOD*, pages 375–388, 2009.

[17] M. Hua, J. Pei, and X. Lin. Ranking queries on uncertain data. *The VLDB Journal*, 20(1):129–153, 2011.

[18] T. Jayram, S. Kale, and E. Vee. Efficient aggregation algorithms for probabilistic data. In *Proc. 18th Annu. ACM-SIAM Sympos. Discrete Alg.*, pages 346–355, 2007.

[19] J. Kleinberg, Y. Rabani, and É. Tardos. Allocating bandwidth for bursty connections. *SIAM Journal on Computing*, 30(1):191–217, 2000.

[20] A. I. Kuksa and N. Z. Šor. The method of estimating the number of conditionally optimal trajectories of discrete separable dynamic programming (in russian). *Kibernetika (Kiev)*, 6:37–44, 1972.

[21] J. Li and H. Wang. Range queries on uncertain data. In *Algorithms and Computation*, pages 326–337. 2014.

[22] X. Lian and L. Chen. Probabilistic top-k dominating queries in uncertain databases. *Information Sciences*, 226:23–46, 2013.

[23] M. Patrascu and L. Roditty. Distance oracles beyond the Thorup-Zwick bound. In *Proc. 51st Annu. IEEE Found. Comp. Sci.*, pages 815–823, 2010.

[24] S. Rahul and R. Janardan. Algorithms for range-skyline queries. In *20th ACM Int. Conf. Adv. Geogr. Inf. Sys.*, 2012.

[25] E. Samuel-Cahn. Comparison of threshold stop rules and maximum for independent nonnegative random variables. *Ann. Probab.*, 12(4):1213–1216, 11 1984.

[26] S. Singh, C. Mayfield, S. Prabhakar, R. Shah, and S. Hambrusch. Indexing uncertain categorical data. In *Proc. 23rd Annu. IEEE Int. Conf. Data Eng.*, pages 616–625, 2007.

[27] M. A. Soliman, I. F. Ilyas, and K. Chen-Chuan Chang. Top-k query processing in uncertain databases. In *Proc. 23rd Annu. IEEE Int. Conf. Data Eng.*, pages 896–905, 2007.

[28] Y. Tao, X. Xiao, and R. Cheng. Range search on multidimensional uncertain data. *ACM Trans. Database Sys.*, 32(3):15, 2007.

[29] G. Xiao, K. Li, K. Li, and X. Zhou. Efficient top-(k, l) range query processing for uncertain data based on multicore architectures. *Distributed & Parallel Databases*, pages 1–33, 2014.

[30] K. Yi, F. Li, G. Kollios, and D. Srivastava. Efficient processing of top-k queries in uncertain databases. In *Proc. 24th Annu. IEEE Int. Conf. Data Eng.*, pages 1406–1408, 2008.

[31] M. L. Yiu, N. Mamoulis, X. Dai, Y. Tao, and M. Vaitis. Efficient evaluation of probabilistic advanced spatial queries on existentially uncertain data. *IEEE Trans. Knowl. Data Eng.*, 21(1):108–122, 2009.