Top-k Querying of Unknown Values under Order Constraints

Antoine Amarilli[1], Yael Amsterdamer[2], Tova Milo[3], and Pierre Senellart[4,5]

1 **LTCI, Télécom ParisTech, Université Paris-Saclay, Paris, France**
`first.last@telecom-paristech.fr`
2 **Bar Ilan University, Ramat Gan, Israel**
`first.last@biu.ac.il`
3 **Tel Aviv University, Tel Aviv, Israel**
`last@cs.tau.ac.il`
4 **DI, École normale supérieure, PSL Research University, Paris, France; and Inria Paris, Paris, France**
`first.last@ens.fr`

Abstract

Many practical scenarios make it necessary to evaluate top-k queries over data items with partially unknown values. This paper considers a setting where the values are taken from a numerical domain, and where some *partial order constraints* are given over known and unknown values: under these constraints, we assume that all possible worlds are equally likely. Our work is the first to propose a principled scheme to derive the value distributions and expected values of unknown items in this setting, with the goal of computing estimated top-k results by interpolating the unknown values from the known ones. We study the complexity of this general task, and show tight complexity bounds, proving that the problem is intractable, but can be tractably approximated. We then consider the case of tree-shaped partial orders, where we show a constructive PTIME solution. We also compare our problem setting to other top-k definitions on uncertain data.

1998 ACM Subject Classification H.2 Database Management

Keywords and phrases uncertainty, partial order, unknown values, crowdsourcing, interpolation

Digital Object Identifier 10.4230/LIPIcs.ICDT.2017.5

1 Introduction

Many data analysis tasks involve queries over ordered data, such as maximum and top-k queries, which must often be evaluated in presence of *unknown data values*. This problem occurs in many real-life scenarios: retrieving or computing exact data values is often expensive, but querying the partially unknown data may still be useful to obtain approximate results, or to decide which data values should be retrieved next. In such contexts, we can often make use of *order constraints* relating the data values, even when they are unknown: for instance, we know that object A is preferred to object B (though we do not know their exact rating).

This paper thus studies the following general problem. We consider a set of numerical values, some of which are unknown, and we assume a *partial order* on these values: we may know that $x \geqslant y$ should hold although the values x or y are unknown. Our goal is to *estimate* the unknown values, in a principled way, and to evaluate top-k queries, namely find the items with (estimated) highest values.

Without further information, one may assume that every valuation compatible with the order constraints is equally likely, i.e., build a probabilistic model where valuations are

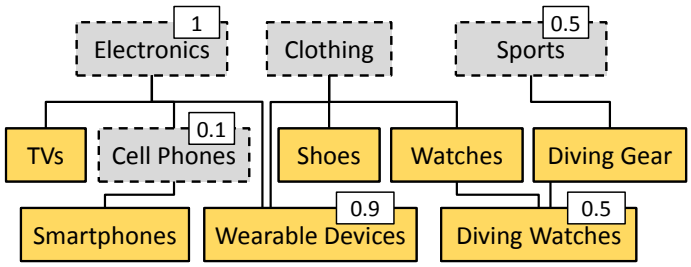


Figure 1 Sample catalog taxonomy with compatibility scores.

uniformly distributed. Indeed, uniform distributions in the absence of prior knowledge is a common assumption in probabilistic data management [1, 12, 34] for continuous distributions on data values within an interval; here we generalize to a uniform distribution over multiple unknown values. Though the distribution is uniform, the dependencies between values lead to non-trivial insights about unknown values and top-k results, as we will illustrate.

Illustrative example. We consider a specific application setting where our problem occurs. Consider a scenario where products are classified in a catalog taxonomy (Figure 1) using human input: the relevance of a product to any category is captured by a *compatibility score*. Assessing this compatibility is often left to human judgement rather than attempting to derive it from records or statistics [49, 9, 40]. Thus we assume scores are obtained via questions to domain experts or to a crowd of unqualified users[1]. Our goal is to assign the product to the top-k most compatible categories among a set of *end categories* (in yellow with a solid border), as opposed to *virtual categories* (dashed border). The virtual categories generalize the end categories, and allow us to ask broader questions to experts, but we do not try to assign products to them, e.g., they do not have a dedicated page in our online store.

Imagine now that the product to classify is a *smartwatch*, and that we want to find the top-2 end categories for it. We asked an expert for its compatibility score with some categories (both end and virtual categories), which we indicate in Figure 1. Because expert input is costly, however, we wish to choose the top-2 end categories based on the incomplete information that we have. The naïve answer is to look only at categories with known scores, and to identify Wearable Devices and Diving Watches as the best end categories.

In this scenario, however, we can impose a natural partial order over the scores, both known and unknown: any product that belongs to a specific category (e.g., Smartphones) conceptually also belongs to each of the more general categories (e.g., Cell Phones). We can thus require that if a category x is a sub-category of y, then the product's compatibility score for x should be at most its score for y. This can be explained in the instructions to the humans providing the scores, and enforced by the user interface. Beyond this constraint, we do not make any further assumption on the scores.

Now, note that due to order constraints, the scores of Watches and Diving Gear, while unknown, cannot be lower than that of Diving Watches; so either of the two could replace Diving Watches in the top-2 answer. To choose between these categories, we observe that the score of Diving Gear must be exactly 0.5 (which bounds it from above and below). In contrast, as the score of Wearable Devices is 0.9, Clothing has a score of at least 0.9, so the score of Watches can be anything between 0.5 and an unknown value which is $\geqslant 0.9$. A

[1] In the latter case, we aggregate the answers of multiple workers to obtain the compatibility score.

better top-2 answer is thus Wearable Devices and Watches, the latter being likely to have a higher score than Diving Watches (or Diving Gear).

Other application domains. Beyond the crowdsourcing example that we illustrate, our setting is relevant to many other application domains involving different types of unknown data. For instance, in the domain of Web services [46], one may wish to find the k-most relevant apartments for a user, given user-provided criteria on the apartment (price range, location, size, etc.) but no precise way to aggregate them into a relevance function. In this case, order constraints may be imposed on the unknown relevance of apartments, whenever one apartment *dominates* another (i.e., is better for each criterion); exact-value constraints may capture user-provided ratings to viewed apartments; and a top-k query could be used, e.g., to select the most relevant apartments among those available for sale.

As another example, in the context of top-k queries over sensor data [25, 35], one may wish to find the k-fastest drivers in a certain region given partial data from speedometers and street cameras; comparing the progress and locations of vehicles may yield partial order constraints on their speed. Other domains include, e.g., data mining [3], managing preference data [48], or finding optimal values of black-box functions expressing cost, running time, etc.

Contributions. As previously mentioned, we assume a uniform probability distribution over valuations of unknown items, which we capture formally in our model via possible-world semantics. We then use the *expected values* of items as an estimate of their unknown values, for the sake of top-k computation (see Section 2). Our work presents three main contributions using this general model, as follows.

First, in Section 3 we present a *general and principled scheme to interpolate unknown values from known ones under partial order constraints*, and thereby obtain the top-k such values. We implement this in an algorithm that is polynomial in the number of possible item orderings, and consequently show that in the worst case it is in $\mathrm{FP}^{\#\mathrm{P}}$ in the size of the input.[2] The problem of finding expected values has a geometric characterization as centroid computation in high-dimensional polytopes (as we explain further); however, our $\mathrm{FP}^{\#\mathrm{P}}$ membership result goes beyond existing computational geometry results since the constraints that we consider, namely, partial order and exact-value constraints, correspond to special classes of polytopes not studied in the context of geometry. Indeed, centroid computation is generally not in $\mathrm{FP}^{\#\mathrm{P}}$ [32, 42]. Our work also departs from previous work on top-k queries over incomplete or probabilistic data [15, 28, 46]: we do not make the simplifying assumption that item distributions are independent and given, but rather study the effect of constraints on individual item distributions.

Our second main contribution, in Section 4, is to establish *hardness results*, and specifically, a matching lower $\mathrm{FP}^{\#\mathrm{P}}$ bound for top-k computation. While the #P-hardness of computing expected values follows from the geometric characterization of the problem [42], we show that top-k is hard even without computing expected values. Hence the $\mathrm{FP}^{\#\mathrm{P}}$ bound is tight for both interpolation and top-k; this shows that the assumption regarding variable independence in previous work, that enables PTIME solutions [15, 28, 46], indeed simplifies the problem. To complete the picture we discuss possible approximation schemes for the interpolation and top-k problems, again by the connection to centroid computation.

[2] #P is the class of counting problems that return the number of solutions of NP problems. $\mathrm{FP}^{\#\mathrm{P}}$ is the class of function problems that can be computed in PTIME using a #P oracle.

Our third main contribution, in Section 5, is the study of *tractable cases*, following the hardness of the general problem and the high complexity of approximation. We devise a PTIME algorithm to compute expected values and top-k results when the order constraints are *tree-shaped* or decomposable to trees. This class of constraints is commonly encountered in the context of unknown values (e.g., taxonomies of products are often trees rather than DAGs); yet, to our knowledge, the corresponding polytopes have no equivalents in the context of computational geometry.

Our results also include a review of existing definitions for top-k over uncertain data, which motivates our particular choice of definition (in Section 6). We survey related work in more depth in Section 7 and conclude in Section 8. Full proofs of our results can be found in [5], an extended version of this paper.

2 Preliminaries and Problem Statement

This section introduces the formal definitions for the problem that we study in this paper. We model known and unknown item values as *variables*, and order constraints as *equalities* and *inequalities* over them. Then we define the possible valuations for the variables via possible-world semantics, and use this semantics to define a uniform distribution where all worlds are equally likely. The problem of top-k querying over unknown values can then be formally defined with respect to the expected values of variables in the resulting distribution.

2.1 Unknown Data Values under Constraints

Our input includes a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ of variables with unknown values $v(x_1), \ldots, v(x_n)$, which we assume[3] to be in the range $[0, 1]$. We consider two kinds of constraints over them:

- **order constraints**, written $x_i \leqslant x_j$ for $x_i, x_j \in \mathcal{X}$, encoding that $v(x_i) \leqslant v(x_j)$;
- **exact-value constraints** to represent variables with known values, written[4] $x_i = \alpha$ for $0 \leqslant \alpha \leqslant 1$ and for $x_i \in \mathcal{X}$, encoding that $v(x_i) = \alpha$.

In what follows, a *constraint set* with constraints of both types is typically denoted $\mathcal{C}$. We assume that constraints in $\mathcal{C}$ are *not contradictory* (e.g., we forbid $x = 0.1$, $y = 0.2$, $y \leqslant z$, and $z \leqslant x$), and that they are *closed under implication*: e.g., if $x = \alpha$, $y = \beta$ are given, and $\alpha \leqslant \beta$, then $x \leqslant y$ is implied and thus should also be in $\mathcal{C}$. We can check in PTIME that $\mathcal{C}$ is non-contradictory by simply verifying that it does not entail a false inequality on exact values (e.g., $0.2 \leqslant 0.1$ as in our previous example). The closure of $\mathcal{C}$ can be found in PTIME as a transitive closure computation [27] that also considers exact-value constraints. We denote by $\mathcal{X}_{\text{exact}}$ the subset of $\mathcal{X}$ formed of variables with exact-value constraints.

▶ **Example 1.** In the product classification example from the Introduction, a variable $x_i \in \mathcal{X}$ would represent the compatibility score of the product to the i-th category. If the score is known, we would encode it as a constraint $x_i = \alpha$. In addition, $\mathcal{C}$ would contain the order constraint $x_i \leqslant x_j$ whenever category i is a sub-category of j (recall that the score of a sub-category cannot be higher than that of an ancestor category).

[3] Our results extend to other bounded, continuous ranges, because we can rescale them to fall in $[0, 1]$.
[4] The number α is written as a rational number, represented by its numerator and denominator.

2.2 Possible World Semantics

The unknown data captured by $\mathcal{X}$ and $\mathcal{C}$ makes infinitely many valuations of $\mathcal{X}$ possible (including the true one). We model these options via possible world semantics: a *possible world* w for a constraint set $\mathcal{C}$ over $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a vector of values $w = (v_1, \ldots, v_n) \in [0,1]^n$, corresponding to setting $v(x_i) := v_i$ for all i, such that all the constraints of $\mathcal{C}$ hold under this valuation. The set of all possible worlds is denoted by $\mathrm{pw}_{\mathcal{X}}(\mathcal{C})$, or by $\mathrm{pw}(\mathcal{C})$ when $\mathcal{X}$ is clear from context.

Notice that $\mathcal{C}$ can be encoded as a set of *linear constraints*, i.e., a set of inequalities between linear expressions on $\mathcal{X}$ and constants in $[0,1]$. Thus, following common practice in linear programming, the feasible region of a set of linear constraints ($\mathrm{pw}(\mathcal{C})$ in our setting) can be characterized geometrically as a convex polytope, termed the *admissible polytope*: writing $n := |\mathcal{X}|$, each linear constraint defines a feasible half-space of $\mathbb{R}^n$ (e.g., the half-space where $x \leqslant y$), and the convex polytope $\mathrm{pw}(\mathcal{C})$ is the intersection of all half-spaces. In our setting the polytope $\mathrm{pw}(\mathcal{C})$ is bounded within $[0,1]^n$, and it is non-empty by our assumption that $\mathcal{C}$ is not contradictory. With exact-value constraints, or order constraints such as $x_i \leqslant x_j$ and $x_j \leqslant x_i$, it may be the case that the dimension of this admissible polytope is less than $|\mathcal{X}|$. Computing this dimension can easily be done in PTIME (see, e.g., [44]).

▶ **Example 2.** Let $\mathcal{X} = \{x, y, z\}$. If $\mathcal{C} = \{x \leqslant y\}$, the admissible polytope has dimension 3 and is bounded by the planes defined by $x = y$, $x = 0$, $y = 1$, $z = 0$ and $z = 1$. If we add to $\mathcal{C}$ the constraint $y = 0.3$, the admissible polytope is a 2-dimensional rectangle bounded by $0 \leqslant x \leqslant 0.3$ and $0 \leqslant z \leqslant 1$ on the $y = 0.3$ plane. We cannot add, for example, the constraint $x = 0.5$, because $\mathcal{C}$ would become contradictory.

2.3 Probability Distribution

Having characterized the possible worlds of $\mathrm{pw}(\mathcal{C})$, we assume a *uniform* probability distribution over $\mathrm{pw}(\mathcal{C})$, as indicated in the Introduction. This captures the case when all possible worlds are equally likely, and is a natural choice when we have no information about which valuations are more probable.

Since the space of possible worlds is continuous, we formally define this distribution via a probability density function (pdf), as follows. Let $\mathcal{X}$ and $\mathcal{C}$ define a d-dimensional polytope $\mathrm{pw}_{\mathcal{X}}(\mathcal{C})$ for some integer d. The *d-volume* (also called the Lebesgue measure [29] on $\mathbb{R}^d$) is a measure for continuous subsets of d-dimensional space, which coincides with length, area, and volume for dimensions 1, 2, and 3, respectively. We denote by $V_d(\mathcal{C})$ the d-volume of the admissible polytope, or simply $V(\mathcal{C})$ when d is the dimension of $\mathrm{pw}(\mathcal{C})$.

▶ **Definition 3.** The *uniform pdf* p maps each possible world $w \in \mathrm{pw}(\mathcal{C})$ to the constant $p(w) := 1/V(\mathcal{C})$.

2.4 Top-k Queries

We are now ready to formally define the main problem studied in this paper, namely, the evaluation of *top-k queries* over unknown data values. The queries that we consider retrieve the k items that are estimated to have the highest values, *along with their estimated values*, with ties broken arbitrarily. We further allow queries to apply a *selection operator* σ on the items before performing the top-k computation. In our example from the Introduction, this is what allows us to select the top-k categories among only the end categories. We denote the subset of $\mathcal{X}$ selected by σ as $\mathcal{X}_\sigma$.

If all item values are known, the semantics of top-k queries is clear. In presence of unknown values, however, the semantics must be redefined to determine how the top-k items and their values are estimated. In this paper, we estimate unknown items by their *expected value over all possible worlds*, i.e., their expected value according to the uniform pdf p defined above on $\mathrm{pw}(\mathcal{C})$. This corresponds to *interpolating* the unknown values from the known ones, and then querying the result. We use these interpolated values to define the top-k problem as computing the k variables with the highest expected values, but we also study on its own the interpolation problem of computing the expected values.

To summarize, the two formal problems that we study on constraint sets are:

Interpolation. Given a constraint set $\mathcal{C}$ over $\mathcal{X}$ and variable $x \in \mathcal{X}$, the *interpolation problem* for x is to compute the expected value of x in the uniform distribution over $\mathrm{pw}_{\mathcal{X}}(\mathcal{C})$.

Top-k. Given a constraint set $\mathcal{C}$ over $\mathcal{X}$, a selection predicate σ, and an integer k, the *top-k computation problem* is to compute the ordered list of the k maximal expected values of variables in $\mathcal{X}_\sigma$ (or less if $|\mathcal{X}_\sigma| \leqslant k$), with ties broken arbitrarily.

We review other definitions of top-k on uncertain data in Section 6, where we justify our choice of semantics.

Alternate phrasing. The Interpolation problem can also be defined geometrically, as the computation of the *centroid* (or *center of mass*) of the admissible polytope: the point G such that all vectors relative to G originating at points within the polytope sum to zero. The constraints that we study correspond to a special kind of polytopes, for which we will design a specific algorithm in the next section, and derive an $\mathrm{FP}^{\#\mathrm{P}}$ membership bound which does not hold for general polytopes (as explained in the Introduction). However, the geometric connection will become useful when we study the complexity of our problem in Section 4.1.

3 An Algorithm for Interpolation and Top-k

Having defined formally the problems that we study, we begin our complexity analysis by designing an algorithm that computes the expected value of variables.

The algorithm enumerates all possible orderings of the variables (to be defined formally below), but it is still nontrivial: we must handle exact-value constraints specifically, and we must compute the probability of each ordering to determine its weight in the overall expected value computation. From the algorithm, we will deduce that our interpolation and top-k problems are in $\mathrm{FP}^{\#\mathrm{P}}$.

Eliminating ties. To simplify our study, we will eliminate from the start the problem of *ties*, which will allow us to assume that values in all worlds are totally ordered. We say that a possible world $w = (v_1, \ldots, v_n)$ of $\mathcal{C}$ has a *tie* if $v_i = v_j$ for some i, j. Note that occasional ties, not enforced by $\mathcal{C}$, have *an overall probability of* 0: intuitively, if the admissible polytope is d-dimensional, then all the worlds where $v_i = v_j$ correspond to a $(d-1)$-dimensional hyperplane bounding or intersecting the polytope. A finite set of such hyperplanes (for every pair of variables) has total d-volume 0. Since our computations (volume, expected value) involve integrating over possible worlds, a set of worlds with total probability 0 does not affect the result.

What is left is to consider ties enforced by $\mathcal{C}$ (and thus having probability 1). In such situations, we can rewrite $\mathcal{C}$ by merging these variables to obtain an equivalent constraint set where ties have probability 0. Formally:

▶ **Lemma 4.** *For any constraint set $\mathcal{C}$, we can construct in PTIME a constraint set $\mathcal{C}'$ such that the probability that the possible worlds of $\mathcal{C}'$ have a tie (under the uniform distribution) is zero, and such that any interpolation or top-k computation problem on $\mathcal{C}$ can be reduced in PTIME to the same type of problem on $\mathcal{C}'$.*

Hence, we assume from now on that ties have zero probability in $\mathcal{C}$, so that we can ignore possible worlds with ties without affecting the correctness of our analysis. Note that this implies that all of our results also hold for *strict inequality constraints*, of the forms $x < y$ and $x \neq y$.

Under this assumption, we first study in Section 3.1 the case where $\mathcal{C}$ is a total order. We then handle arbitrary $\mathcal{C}$ by aggregating over possible variable orderings, in Section 3.2.

3.1 Total Orders

In this section we assume $\mathcal{C}$ is a total order $\mathcal{C}_1^n(\alpha, \beta)$ defined as $x_0 \leqslant x_1 \leqslant \cdots \leqslant x_n \leqslant x_{n+1}$, where $x_0 = \alpha$ and $x_{n+1} = \beta$ are variables with exact-value constraints in $\mathcal{X}_{\text{exact}}$.

We first consider *unfragmented* total orders, where $x_1, \ldots, x_n \notin \mathcal{X}_{\text{exact}}$. In this case, we can show that the expected value of x_i, for $1 \leqslant i \leqslant n$, corresponds to a *linear interpolation* of the unknown variables between α and β, namely: $\frac{i}{n+1} \cdot (\beta - \alpha) + \alpha$. This can be shown formally via a connection to the expected value of the order statistics of samples from a uniform distribution, which follows a Beta distribution [22].

Now consider the case of *fragmented* total orders, where $\mathcal{C}$ is allowed to contain more exact-value constraints than the ones on α and β. We observe that we can *split* the total order into *fragments*: by cutting at each variable that has an exact-value constraint, we obtain sub-sequences of variables which follow an unfragmented total order. We can then compute the expected values of each fragment independently, and compute the total order volume as the product of the fragment volumes. The correctness of this computation follows from a more general result (Lemma 12) stated and proven in Section 5.

Hence, given a constraint set $\mathcal{C}$ imposing a (possibly fragmented) total order, the expected value of x_i can be computed as follows. If $x_i \in \mathcal{X}_{\text{exact}}$, analysis is trivial. Otherwise, we consider the fragment $\mathcal{C}_{p+1}^{q-1}(v_p, v_q)$ that contains x_i; namely, p is the maximal index such that $0 \leqslant p < i$ and $x_p \in \mathcal{X}_{\text{exact}}$, and q is the minimal index such that $i < q \leqslant n+1$ and $x_q \in \mathcal{X}_{\text{exact}}$. The expected value of x_i can then be computed within $\mathcal{C}_{p+1}^{q-1}(v_p, v_q)$ using linear interpolation.

The following proposition summarizes our findings:

▶ **Proposition 5.** *Given a constraint set $\mathcal{C}$ implying a total order, the expected value of any variable $x_i \in \mathcal{X}$ can be computed in PTIME.*

3.2 General Constraint Sets

We can now extend the result for total orders to an expression of the expected value for a general constraint set $\mathcal{C}$. We apply the previous process to each possible total ordering of the variables, and aggregate the results. To do this, we define the notion of *linear extensions*, inspired by partial order theory:

▶ **Definition 6.** Given a constraint set $\mathcal{C}$ over $\mathcal{X}$, we say that a constraint set $\mathcal{T}$ is a *linear extension* of $\mathcal{C}$ if (i) $\mathcal{T}$ is a total order; (ii) the exact-value constraints of $\mathcal{T}$ are exactly those of $\mathcal{C}$; and (iii) $\mathcal{C} \subseteq \mathcal{T}$, namely every constraint $x \leqslant y$ in $\mathcal{C}$ also holds[5] in $\mathcal{T}$.

[5] The linear extensions of $\mathcal{C}$ in this sense are thus exactly the linear extensions of the partial order on $\mathcal{X}$ imposed by $\mathcal{C}$: this partial order is indeed antisymmetric because $\mathcal{C}$ has no ties.

Algorithm 1: Compute the expected value of a variable.

Input: Constraint set $\mathcal{C}$ on variables $\mathcal{X}$ with $n := |\mathcal{X}|$ where ties have probability 0 and where value range $[0, 1]$ is enforced by constraints; variable $x \in \mathcal{X}$

Output: Expected value of x

1 **if** *x is in $\mathcal{X}_{\text{exact}}$, i.e., has an exact-value constraint in $\mathcal{C}$ to some v* **then return** v;
2 $\mathbb{E}_{\mathcal{C}}[x] \leftarrow 0$; $V(\mathcal{C}) \leftarrow 0$;
3 $m \leftarrow |\mathcal{X}_{\text{exact}}|$;
4 Write $x_0 < \cdots < x_{m-1}$ the variables of $\mathcal{X}_{\text{exact}}$ and $v_0 < \cdots < v_{m-1}$ their values;
5 **foreach** *linear extension $\mathcal{T}$ of $\mathcal{C}$* **do**
6 Write $i_0 < \cdots < i_{m-1}$ the *indices* in $\mathcal{T}$ of variables $x_0, \ldots, x_{m-1}$ in $\mathcal{X}_{\text{exact}}$;
7 Write k the *index* in $\mathcal{T}$ of the variable x;
8 Write i_j, i_{j+1} the indices of variables from $\mathcal{X}_{\text{exact}}$ s.t. $i_j < k < i_{j+1}$;
9 $\mathbb{E}_{\mathcal{T}}[x] \leftarrow \text{ExpectedValFrag}(i_j, i_{j+1}, k, v_j, v_{j+1})$; `// Expected value of` x `in` $\mathcal{T}$
10 $V(\mathcal{T}) \leftarrow \prod_{l=0}^{m-2} \text{VolumeFrag}(i_l, i_{l+1}, v_l, v_{l+1})$; `// Volume of` $\mathcal{T}$
11 $\mathbb{E}_{\mathcal{C}}[x] \leftarrow \mathbb{E}_{\mathcal{C}}[x] + V(\mathcal{T}) \times \mathbb{E}_{\mathcal{T}}[x]$; `// Sum exp. value of` x `weighted by` $V(\mathcal{T})$
12 $V(\mathcal{C}) \leftarrow V(\mathcal{C}) + V(\mathcal{T})$; `// Sum of total order volumes`
13 **return** $\frac{\mathbb{E}_{\mathcal{C}}[x]}{V(\mathcal{C})}$;
14 **Function** ExpectedValFrag(p, q, k, α, β)
Input: $p < q$: indices; k: requested variable index; $\alpha < \beta$: exact values at p, q resp.
Output: Expected value of variable x_k in the fragment $\mathcal{C}_{p+1}^{q-1}(\alpha, \beta)$
15 $n \leftarrow q - p - 1$; `// num. of variables in the fragment which are` $\notin \mathcal{X}_{\text{exact}}$
16 **return** $\frac{k-p}{n+1} \cdot (\beta - \alpha) + \alpha$; `// linear interpolation (see Section 3.1)`
17 **Function** VolumeFrag(p, q, α, β)
Input: $p < q$: indices; $\alpha < \beta$: exact values at p, q resp.
Output: $V(\mathcal{C}_{p+1}^{q-1}(\alpha, \beta))$: volume of the total order fragment between indices p, q
18 $n \leftarrow q - p - 1$; `// num. of variables in the fragment which are` $\notin \mathcal{X}_{\text{exact}}$
19 **return** $\frac{(\beta-\alpha)^n}{n!}$; `// Volume of` $[\alpha, \beta]^n$ `divided by num. of total orders`

Algorithm 1 presents our general scheme to compute the expected value of a variable $x \in \mathcal{X}$ under an arbitrary constraint set $\mathcal{C}$, assuming the uniform distribution on $\text{pw}(\mathcal{C})$.

The algorithm iterates over each linear extension $\mathcal{T}$ of $\mathcal{C}$, and computes the expected value of x in $\mathcal{T}$ and the overall probability of $\mathcal{T}$ in $\text{pw}(\mathcal{C})$. A linear extension is a total order, so x is within a particular fragment of it, namely, between the indices of two consecutive variables with exact-value constraints, i_j and i_{j+1}. The *expected value of x in $\mathcal{T}$*, denoted by $\mathbb{E}_{\mathcal{T}}[x]$, is then affected only by the constraints and variables of this fragment, and can be computed using linear interpolation by the function ExpectedValFrag (line 9).

Now, the final expected value of x in $\mathcal{C}$ is the average of all $\mathbb{E}_{\mathcal{T}}[x]$ weighted by the probability of each linear extension $\mathcal{T}$, i.e., the volume of $\text{pw}(\mathcal{T})$ divided by the volume of $\text{pw}(\mathcal{C})$. Recall that, by Lemma 4, worlds with ties have total volume 0 and do not affect this expected value. We compute the volume of $\mathcal{T}$ as the *product of volumes of its fragments* (line 10). The volume of a fragment, computed by function VolumeFrag, is the volume of $[\alpha, \beta]^n$, i.e., all assignments to the n variables of the fragment in $[\alpha, \beta]$, divided by the number of orderings of these variables, to obtain the volume of one specific order (line 19).

The complexity of Algorithm 1 is polynomial in the number of linear extensions of $\mathcal{C}$, as we can enumerate them in constant amortized time [41]. However, in the general case, there may be up to $|\mathcal{X}|!$ linear extensions. To obtain an upper bound in the general case, we note that

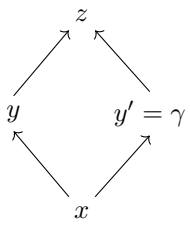
we can rescale all constraints so that all numbers are integers, and then nondeterministically sum over the linear extensions. This yields our $\mathrm{FP}^{\#\mathrm{P}}$ upper bound:

▶ **Theorem 7.** *Given a constraint set $\mathcal{C}$ over $\mathcal{X}$ and $x \in \mathcal{X}$ (resp., and a selection predicate σ, and an integer k), determining the expected value of x in $\mathrm{pw}(\mathcal{C})$ under the uniform distribution (resp., the top-k computation problem over $\mathcal{X}$, $\mathcal{C}$, and σ) is in $\mathrm{FP}^{\#\mathrm{P}}$.*

The $\mathrm{FP}^{\#\mathrm{P}}$ membership for interpolation does not extend to centroid computation in general convex polytopes, which is not in $\mathrm{FP}^{\#\mathrm{P}}$ [32, 42]. Our algorithm thus relies on the fact that the polytope $\mathrm{pw}(\mathcal{C})$ is of a specific form, defined with order and exact-value constraints. The same upper bound for the top-k problem immediately follows. We will show in Section 4 that this $\mathrm{FP}^{\#\mathrm{P}}$ upper bound is tight.

We also provide a complete example to illustrate the constructions of this section.

Full Example. We exemplify our scheme on variables $\mathcal{X} = \{x, y, y', z\}$ and on the constraint set $\mathcal{C}$ generated by the order constraints $x \leqslant y$, $y \leqslant z$, $x \leqslant y'$, $y' \leqslant z$ and the exact-value constraint $y' = \gamma$ for some fixed $0 < \gamma < 1$. Remember that we necessarily have $0 \leqslant x$ and $z \leqslant 1$ as well. The constraints of $\mathcal{C}$ are closed under implication, so they also include $x \leqslant z$. The figure shows the Hasse diagram of the partial order defined by $\mathcal{C}$ on $\mathcal{X}$. Note that ties have a probability of zero in $\mathrm{pw}(\mathcal{C})$.



The two linear extensions of $\mathcal{C}$ are $\mathcal{T}_1 : x \leqslant y \leqslant y' \leqslant z$ and $\mathcal{T}_2 : x \leqslant y' \leqslant y \leqslant z$. Now, $\mathcal{T}_1$ is a fragmented total order, and we have $\mathrm{pw}(\mathcal{T}_1) = \mathrm{pw}_{\{x,y\}}(\mathcal{C}') \times \{\gamma\} \times [\gamma, 1]$ where $\mathcal{C}'$ is defined on variables $\{x, y\}$ by $0 \leqslant x \leqslant y \leqslant \gamma$. We can compute the volume of $\mathrm{pw}(\mathcal{T}_1)$ as $\alpha_1 = \frac{\gamma^2}{2} \times (1 - \gamma)$. Similarly the volume of $\mathrm{pw}(\mathcal{T}_2)$ is $\alpha_2 = \gamma \times \frac{(1-\gamma)^2}{2}$.

Let us compute the expected value of y for $\mathcal{C}$. In $\mathcal{T}_1$ its expected value is $\mu_1 = \mathbb{E}_{\mathcal{T}_1}[y] = \frac{2}{3} \cdot (\gamma - 0) + 0 = \frac{2}{3}\gamma$. In $\mathcal{T}_2$ its expected value is $\mu_2 = \mathbb{E}_{\mathcal{T}_2}[y] = \frac{1}{3} \cdot (1 - \gamma) + \gamma = \frac{1+2\gamma}{3}$. The overall expected value of y is the average of these expected values weighted by total order probabilities (volumes fractions), namely $\mathbb{E}_{\mathcal{C}}[y] = \frac{\alpha_1\mu_1 + \alpha_2\mu_2}{\alpha_1 + \alpha_2}$.

4 Hardness and Approximations

We next show that the intractability of Algorithm 1 in Section 3 is probably unavoidable. We first show matching lower bounds for interpolation and top-k in Section 4.1. We then turn in Section 4.2 to the problem of approximating expected values.

4.1 Hardness of Exact Computation

We now analyze the complexity of computing an exact solution to our two main problems. We show below a new result for the hardness of top-k. But first, we state the lower bound for the interpolation problem, which is obtained via the geometric characterization of the problem. In previous work, centroid computation is proven to be hard for *order polytopes*, namely, polytopes without exact-value constraints, which are a particular case of our setting:

▶ **Theorem 8. ([42], Theorem 1).** *Given a set $\mathcal{C}$ of order constraints and $x \in \mathcal{X}$, determining the expected value of x in $\mathrm{pw}(\mathcal{C})$ under the uniform distribution is $\mathrm{FP}^{\#\mathrm{P}}$-hard.*

We now show a new lower bound for top-k queries: interestingly, these queries are $\mathrm{FP}^{\#\mathrm{P}}$-hard *even if they do not have to return the expected values*. Recall that σ is the selection operator (see Section 2.4), which we use to compute top-k among a restricted subset

of variables. We can show hardness even for top-1 queries, and even when σ only selects two variables:

▶ **Theorem 9.** *Given a constraint set $\mathcal{C}$ over $\mathcal{X}$, a selection predicate σ, and an integer k, the top-k computation problem over $\mathcal{X}$, $\mathcal{C}$ and σ is $\mathrm{FP}^{\#\mathrm{P}}$-hard even if k is fixed to be 1, $|\mathcal{X}_\sigma|$ is 2, and the top-k answer does not include the expected value of the variables.*

Proof sketch. To prove hardness in this case, we reduce from interpolation. We show that a top-1 computation oracle can be used as a comparison oracle to compare the expected value of a variable x to any other rational value α, by adding a fresh element x' with an exact-value constraint to α and using σ to compute the top-1 among $\{x, x'\}$. What is more technical is to show that, given such a comparison oracle, we can perform the reduction and determine *exactly* the expected value v of x (a rational number) using only a *polynomial* number of comparisons to other rationals. This follows from a bound on the denominator of v, and by applying the rational number identification scheme of [39]. ◀

In settings where we do not have a selection operator (i.e., $\mathcal{X}_\sigma = \mathcal{X}$), we can similarly show the hardness of top-k (rather than top-1). See [5] for details.

4.2 Complexity of Approximate Computation

In light of the previous hardness results, we now review approximation algorithms, again via the geometric characterization of our setting. In Section 5, we will show a novel exact solution in PTIME for specific cases.

The *interpolation* problem can be shown to admit a fully polynomial-time randomized approximation scheme (FPRAS). This result follows from existing work [31, 7], using a tractable almost uniform sampling scheme for convex bodies.

▶ **Proposition 10** ([31], Algorithm 5.8)**.** *Let $\mathcal{C}$ be a set of constraints with variable set $\mathcal{X}$ and $x \in \mathcal{X}$. There is an FPRAS that determines an estimate $\hat{\mathbb{E}}_{\mathcal{C}}[x]$ of the expected value $\mathbb{E}_{\mathcal{C}}[x]$ of x in $\mathrm{pw}(\mathcal{C})$ under the uniform distribution.*

This result is mostly of theoretical interest, as the polynomial is in $|\mathcal{X}|^7$ (see [7], Table 1), but recent improved sampling algorithms [37] may ultimately yield a practical approximate interpolation technique for general constraint sets (see [36, 21]).

For completeness, we mention two natural ways to define randomized approximations for *top-k computation*:

- We can define the *approximate top-k* as an ordered list of k items whose expected value does not differ by more than some $\epsilon > 0$ from that of the item in the actual top-k at the same rank. An FPRAS for this definition of approximate top-k can be obtained from that of Proposition 10.
- It is highly unlikely that there exists a PTIME algorithm to return the *actual top-k* with high probability, even without requiring it to return the expected values. Indeed, such an algorithm would be in the BPP (bounded-error probabilistic time) complexity class; yet it follows from Theorem 9 above that deciding whether a set of variables is the top-k is NP-hard, so the existence of the algorithm would entail that NP $\subseteq$ BPP.

5 Tractable Cases

Given the hardness results in the previous section and the impracticality of approximation, we now study whether *exact* interpolation and top-k computation can be tractable on restricted classes of constraint sets. We consider tree-shaped constraints (defined formally below) and generalizations thereof: they are relevant for practical applications (e.g., classifying items into tree- or forest-shaped taxonomies), and we will show that our problems are tractable on them. We start by a splitting lemma to decompose constraint sets into "independent" subsets of variables, and then define and study our tractable class.

5.1 Splitting Lemma

We will formalize the cases in which the valuations of two variables in $\mathcal{X}$ are probabilistically dependent (the variables *influence* each other), according to $\mathcal{C}$. This, in turn, will enable us to define *independent subsets of the variables* and thus *independent subsets of the constraints* over these variables. This abstract result will generalize the notion of fragments from total orders (see Section 3.1) to general constraint sets. In what follows, we use $x_i \prec x_j$ to denote the *covering relation* of the partial order $\leqslant$, i.e., $x_i \leqslant x_j$ is in $\mathcal{C}$ but there exists no $x_k \notin \{x_i, x_j\}$ such that $x_i \leqslant x_k$ and $x_k \leqslant x_j$ are in $\mathcal{C}$.

▶ **Definition 11.** We define the *influence relation* $x \leftrightarrow y$ between variables of $\mathcal{X}\backslash\mathcal{X}_{\text{exact}}$ as the equivalence relation obtained by the symmetric, reflexive, and transitive closure of the $\prec$ relation on $\mathcal{X}\backslash\mathcal{X}_{\text{exact}}$.

The *uninfluenced classes* of $\mathcal{X}$ under $\mathcal{C}$ is the partition of $\mathcal{X}\backslash\mathcal{X}_{\text{exact}}$ as the subsets $\mathsf{X}_1, \ldots, \mathsf{X}_m$ given by the equivalence classes of the influence relation.

The *uninfluence decomposition of* $\mathcal{C}$ is the collection of constraint sets $\mathcal{C}_1, \ldots, \mathcal{C}_m$ of $\mathcal{C}$ where each $\mathcal{C}_i$ has as variables $\mathsf{X}_i \sqcup \mathcal{X}_{\text{exact}}$ and contains all exact-value constraints of $\mathcal{C}$ and all order constraints between variables of $\mathsf{X}_i \sqcup \mathcal{X}_{\text{exact}}$.

We assume w.l.o.g. that $m > 0$, i.e., there are unknown variables in $\mathcal{X}\backslash\mathcal{X}_{\text{exact}}$; otherwise the uninfluence decomposition is meaningless but any analysis is trivial. Intuitively, two unknown variables x, x' are in different uninfluenced classes if in *every* linear extension there is *some* variable from $\mathcal{X}_{\text{exact}}$ between them, or if they belong to disconnected (and thus incomparable) parts of the partial order. In particular, uninfluenced classes correspond to the fragments of a total order: this is used in Section 3.1. The uninfluence decomposition captures only constraints between *variables that influence each other*, and constraints that can *bound the range of a variable* by making it comparable to variables from $\mathcal{X}_{\text{exact}}$. We formally prove the independence of $\mathcal{C}_1, \ldots, \mathcal{C}_m$ via possible-world semantics: every possible world of $\mathcal{C}$ can be decomposed to possible worlds of $\mathcal{C}_1, \ldots, \mathcal{C}_m$, and vice versa.

▶ **Lemma 12.** *Let $\mathcal{C}_1, \ldots, \mathcal{C}_m$ be the uninfluence decomposition of $\mathcal{C}$. There exists a bijective correspondence between* $\text{pw}(\mathcal{C})$ *and* $\text{pw}(\mathcal{C}_1) \times \cdots \times \text{pw}(\mathcal{C}_m)$.

▶ **Example 13.** Let $\mathcal{X}$ be $\{x, y, y', z, w\}$, and let $\mathcal{C}$ be defined by $y' = 0.5$ and $x \leqslant y \leqslant y' \leqslant z$. The uninfluence classes are $\mathsf{X}_1 = \{x, y\}$, $\mathsf{X}_2 = \{z\}$, and $\mathsf{X}_3 = \{w\}$. The uninfluence decomposition thus consists of $\mathcal{C}_1$, with variables $\mathsf{X}_1 \sqcup \{y'\}$, and constraints $x \leqslant y \leqslant y'$ and $y' = 0.5$; $\mathcal{C}_2$, with variables $\mathsf{X}_2 \sqcup \{y'\}$, and constraints $y' \leqslant z$ and $y' = 0.5$; and $\mathcal{C}_3$, with variables $\mathsf{X}_3 \sqcup \{y'\}$, and constraint $y' = 0.5$.

We next use this independence property to analyse restricted classes of constraint sets.

5.2 Tree-Shaped Constraints

We define the first restricted class of constraints that we consider: *tree-shaped* constraints. Recall that a *Hasse diagram* is a representation of a partial order as a directed acyclic graph, whose nodes correspond to $\mathcal{X}$ and where there is an edge (x, y) if $x \prec y$. An example of such a diagram is the one used in Section 3.2.

▶ **Definition 14.** A constraint set $\mathcal{C}$ over $\mathcal{X}$ is *tree-shaped* if the probability of ties is zero, the Hasse diagram of the partial order induced on $\mathcal{X}$ by $\mathcal{C}$ is a directed tree, the root has exactly one child, and exactly the root and leaves are in $\mathcal{X}_{\text{exact}}$. Thus, $\mathcal{C}$ imposes a global minimal value, and maximal values at each leaf, and no other exact-value constraint.

We call $\mathcal{C}$ *reverse-tree-shaped* if the reverse of the Hasse diagram (obtained by reversing the direction of the edges) is tree-shaped.

Tree-shaped constraints are often encountered in practice, in particular in the context of product taxonomies. Indeed, while our example from Figure 1 is a DAG, many real-life taxonomies are trees: in particular, the Google Product Taxonomy [23] and ACM CCS [2].

We now show that for a tree-shaped constraint set $\mathcal{C}$, unlike the general case, we can tractably compute exact expressions of the expected values of variables. In the next two results, we assume arithmetic operations on rationals to have unit cost, e.g., they are performed up to a fixed numerical precision. Otherwise, the complexities remain polynomial but the degrees may be larger. We first show:

▶ **Theorem 15.** *For any tree-shaped constraint set $\mathcal{C}$ over $\mathcal{X}$, we can compute its volume $V(\mathcal{C})$ in time $O(|\mathcal{X}|^2)$.*

Proof sketch. We process the tree bottom-up, propagating a piecewise-polynomial function expressing the volume of the subpolytope on the subtree rooted at each node as a function of the value of the parent node: we compute it using Lemma 12 from the child nodes. ◀

This result can be applied to prove the tractability of computing the marginal distribution of any variable $x \in \mathcal{X} \backslash \mathcal{X}_{\text{exact}}$ in a tree-shaped constraint set, which is defined as the pdf $p_x(v) := V_{d-1}(\mathcal{C} \cup \{x = v\})/V_d(\mathcal{C})$, where d is the dimension of $\text{pw}(\mathcal{C})$:

▶ **Theorem 16.** *For any tree-shaped constraint set $\mathcal{C}$ on variable set $\mathcal{X}$, for any variable $x \in \mathcal{X} \backslash \mathcal{X}_{\text{exact}}$, the marginal distribution for x is piecewise polynomial and can be computed in time $O(|\mathcal{X}_{\text{exact}}| \times |\mathcal{X}|^2)$.*

Proof sketch. We proceed similarly to the proof of Theorem 15 but with two functions: one for x and its descendants, and one for all other nodes. The additional $|\mathcal{X}_{\text{exact}}|$ factor is because the second function depends on how the value given to x compares to the leaves. ◀

We last deduce that our results for tree-shaped constraints extend to a more general tractable case: constraint sets $\mathcal{C}$ whose uninfluence decomposition $\mathcal{C}_1, \ldots, \mathcal{C}_m$ is such that every $\mathcal{C}_i$ is (reverse-)tree-shaped. By Lemma 12, each $\mathcal{C}_i$ (and its variables) can be considered independently, and reverse-tree-shaped trees can be easily transformed into tree-shaped ones. Our previous algorithms thus apply to this general case, by executing them on each constraint set of the uninfluence decomposition that is relevant to the task (namely, containing the variable x to interpolate, or top-k candidates from the selected variables $\mathcal{X}_\sigma$):

▶ **Corollary 17.** *Given any constraint set $\mathcal{C}$ and its uninfluence decomposition $\mathcal{C}_1, \ldots, \mathcal{C}_m$, assuming that each $\mathcal{C}_i$ is a (reverse-)tree-shaped constraint set, we can solve the interpolation problem in time $O(\max_i |\mathsf{X}_i|^3)$ and the top-k problem in PTIME.*

On large tree-shaped taxonomies (e.g., the Google Product Taxonomy [23]), in an interactive setting where we may ask user queries (e.g., the one in the Introduction), we can improve running times by asking more queries. Indeed, each answer about a category adds an exact-value constraint, and reduces the size of the constraint sets of the uninfluence decomposition, which decreases the overall running time, thanks to the superadditivity of $x \mapsto x^3$. We do not study which variables should be queried in order to reduce the running time of the algorithm; see, e.g., [40] for tree-partitioning algorithms.

6 Other Variants

We have defined top-k computation on constraint sets by considering the *expected value* of each variable under the uniform distribution. Comparing to different definitions of top-k on unknown values that have been studied in previous work, our definition has some important properties [15]: it provides a ready estimation for unknown values (namely, their expected value) and guarantees an output of size k. Moreover, it satisfies the *containment property* of [15], defined in our setting as follows:

▶ **Definition 18.** A top-k definition satisfies the *containment property* if for any constraint set $\mathcal{C}$ on variables $\mathcal{X}$, for any predicate σ (where we write $\mathcal{X}_\sigma$ the selected variables), and for any $k < |\mathcal{X}_\sigma|$, letting S_k and S_{k+1} be the ordered lists of top-k and top-$(k+1)$ variables, S_k is a strict prefix of S_{k+1}.

The containment property is a natural desideratum: computing the top-k for some $k \in \mathbb{N}$ should not give different variables or order for the top-k' with $k' < k$. Our definition clearly satisfies the containment property (except in the case of ties). By contrast, we will now review prominent definitions of top-k on uncertain data from related work [47, 15, 52], and show that they do not satisfy the containment property when we apply them to the possible world distributions studied in our setting. We focus on two prominent definitions, *U-top-k* and *global-top-k* and call our own definition *local-top-k* when comparing to them; we also discuss other variants in [5].

U-top-k. The *U-top-k* variant does not study *individual* variables but defines the output as the *sequence* of k variables most likely to be the top-k (in that order), for the uniform distribution on $\mathrm{pw}(\mathcal{C})$. We call this alternative definition *U-top-k* by analogy with [47, 15]. Interestingly, the U-top-k and local-top-k definitions sometimes disagree in our setting:

▶ **Lemma 19.** *There is a constraint set $\mathcal{C}$ and selection predicate σ such that local-top-k and U-top-k do not match, even for $k = 1$ and without returning expected values or probabilities.*

We can easily design an algorithm to compute U-top-k in PSPACE and in polynomial time in the number of linear extensions of $\mathcal{C}$: compute the probability of each linear extension as in Algorithm 1, and then sum on linear extensions depending on which top-k sequence they realize (on the variables selected by σ), to obtain the probability of each answer. Hence:

▶ **Proposition 20.** *For any constraint set $\mathcal{C}$ over $\mathcal{X}$, integer k and selection predicate σ, the U-top-k query for $\mathcal{C}$ and σ can be computed in PSPACE and in time $\mathrm{O}(\mathrm{poly}(N))$, where N is the number of linear extensions of $\mathcal{C}$.*

Unlike Theorem 7, however, this does not imply $\mathrm{FP}^{\#\mathrm{P}}$-membership: when selecting the most probable sequence, the number of candidate sequences may not be polynomial (as k is not fixed). We leave to future work an investigation of the precise complexity of U-top-k.

We show that in our setting U-top-k does not satisfy the *containment property* of [15].

▶ **Lemma 21.** *There is a constraint set $\mathcal{C}$ without ties such that U-top-k does not satisfy the containment property for the uniform distribution on* $\mathrm{pw}(\mathcal{C})$.

Global-top-k. We now study the *global-top-k* definition [52], and show that it does not respect the containment property either, even though it is defined on individual variables:

▶ **Definition 22.** The *global-top-k query*, for a constraint set $\mathcal{C}$, selection predicate σ, and integer k, returns the k variables that have the highest probability in the uniform distribution on $\mathrm{pw}(\mathcal{C})$ to be among the variables with the k highest values, sorted by decreasing probability.

▶ **Lemma 23.** *There is a constraint set $\mathcal{C}$ without ties such that global-top-k does not satisfy the containment property for the uniform distribution on* $\mathrm{pw}(\mathcal{C})$.

7 Related Work

We extend the discussion about related work from the Introduction.

Ranking queries over uncertain databases. A vast body of work has focused on providing semantics and evaluation methods for order queries over uncertain databases, including top-k and ranking queries (e.g., [15, 19, 25, 26, 28, 33, 43, 46, 50, 51]). Such works consider two main uncertainty types: *tuple-level uncertainty*, where the existence of tuples (i.e., variables) is uncertain, and hence affects the query results [15, 19, 26, 28, 33, 43, 50, 51]; and *attribute-level uncertainty*, more relevant to our problem, where the data tuples are known but some of their values are unknown or uncertain [15, 25, 28, 46]. Top-k queries over uncertain data following [46] was recently applied to crowdsourcing applications in [13]. These studies are relevant to our work as they identify multiple possible semantics for order queries in presence of uncertainty, and specify desired properties for such semantics [15, 28]; our definition of top-k satisfies the desiderata that are relevant to attribute-level uncertainty [28].

We depart from this existing work in two main respects. First, existing work assumes that each variable is given with an *independent function* that describes its probability distribution. We do not assume this, and instead *derive* expressions for the expected values of variables in a principled way from a uniform prior on the possible worlds. Our work is thus well-suited to the many situations where probability distributions on variables are not known, or where they are not independent (e.g., when order constraints are imposed on them). For this reason, the problems that we consider are generally computationally harder. For instance, [46] is perhaps the closest to our work, since they consider the total orders compatible with given partial order constraints. However, they assume independent marginal distributions, so they can evaluate top-k queries by only considering k-sized prefixes of the linear extensions; in our setting even computing the top-1 element is hard (Theorem 9).

The second key difference is that other works *do not try to estimate the top-k values*, because they assume that the marginal distribution is given: they only focus on ranks. In our context, we need to compute missing values, and need to account, e.g., for exact-value constraints and their effect on the probability of possible worlds and on expected values (Section 3).

We also mention our previous work [4] which considers the estimation of uncertain values (expectation and variance), but only in a *total order*, and did not consider complexity issues.

Partial order search. Another relevant research topic, *partial order search*, considers queries over elements in a partially ordered set to find a subset of elements with a certain property [3,

17, 20, 24, 40]. This relates to many applications, e.g., crowd-assisted graph search [40], frequent itemset mining with the crowd [3], and knowledge discovery, where the unknown data is queried via oracle calls [24]. These studies are *complementary to ours*: when the target function can be phrased as a top-k or interpolation problem, if the search is stopped before all values are known, we can use our method to estimate the complete output.

Computational geometry. Our work reformulates the interpolation problem as a centroid computation problem in the polytope of possible worlds defined by the constraint set. This problem has been studied independently by computational geometry work [42, 31, 38].

Computational geometry mostly studies *arbitrary* convex polytopes (corresponding to polytopes defined by arbitrary linear constraint sets), and often considers the task of *volume computation*, which is related to the problem of computing the centroid [42]. In this context, it is known that computing the exact volume of a polytope is not in $\mathrm{FP}^{\#\mathrm{P}}$ because the output is generally not of polynomial size [32]. Nevertheless, several (generally exponential) methods for exact volume computation [11] have been developed. The problem of approximation has also been studied, both theoretically and practically [31, 45, 18, 16, 36, 21]. Our problem of *centroid* computation is studied in [38], whose algorithm is based on the idea of computing the volume of a polytope by computing the lower-dimensional volume of its facets. This is different from our algorithm, which divides the polytope along linear extensions into subpolytopes, for which we apply a specific volume and centroid computation method.

Some works in computational geometry specifically study *order polytopes*, i.e., the polytopes defined by constraint sets with only order constraints and no exact-value constraints. For such polytopes, volume computation is known to be $\mathrm{FP}^{\#\mathrm{P}}$-complete [10], leading to a $\mathrm{FP}^{\#\mathrm{P}}$-hardness result for centroid computation [42]. However, these results do not apply to *exact-value constraints*, i.e., when order polytopes can only express order relations, between variables which are in $[0, 1]$. Exact-value constraints are both highly relevant in practice (to represent numerical bounds, or known information, e.g., for crowdsourcing), allow for more general polytopes, and complicate the design of Algorithm 1, which must perform volume computation and interpolation in each *fragmented* linear order.

Furthermore, to our knowledge, computational geometry works do not study the top-k problem, or polytopes that correspond to tree-shaped constraint sets, since these have no clear geometric interpretation.

Tree-shaped partial orders. Our analysis of tractable schemes for tree-shaped partial orders is reminiscent of the well-known tractability of probabilistic inference in tree-shaped graphical models [8], and of the tractability of probabilistic query evaluation on trees [14] and treelike instances [6]. However, we study continuous distributions on numerical values, and the influence between variables when we interpolate does not simply follow the tree structure; so our results do not seem to follow from these settings.

8 Conclusion

In this paper, we have studied the problems of top-k computation and interpolation for data with unknown values and order constraints. We have provided foundational solutions, including a general computation scheme, complexity bounds, and analysis of tractable cases.

One natural direction for future work is to study whether our tractable cases (tree-shaped orders, sampling) can be covered by more efficient PTIME algorithms, or whether more general tractable cases can be identified: for instance, a natural direction to study would be

partial orders with a *bounded-treewidth* Hasse diagram, following recent tractability results for the related problem of linear extension counting [30]. Another question is to extend our scheme to request additional values from the crowd, as in [3, 13], and reduce the expected error on the interpolated values or top-k query, relative to a user goal. In such a setting, how should we choose which values to retrieve, and could we update incrementally the results of interpolation when we receive new exact-value constraints? Finally, it would be interesting to study whether our results generalize to different prior distributions on the polytope.

Acknowledgements This work is partially supported by the European Research Council under the FP7, ERC grant MoDaS, agreement 291071, by a grant from the Blavatnik Interdisciplinary Cyber Research Center, by the Israel Science Foundation (grant No. 1157/16), and by the Télécom ParisTech Research Chair on Big Data and Market Insights.

References

1 Serge Abiteboul, T-H. Hubert Chan, Evgeny Kharlamov, Werner Nutt, and Pierre Senellart. Capturing continuous data and answering aggregate queries in probabilistic XML. *TODS*, 36(4), 2011.

2 ACM Computing Classification System, 2012. `https://www.acm.org/about/class/class/2012`.

3 Antoine Amarilli, Yael Amsterdamer, and Tova Milo. On the complexity of mining itemsets from the crowd using taxonomies. In *ICDT*, 2014. `doi:10.5441/002/icdt.2014.06`.

4 Antoine Amarilli, Yael Amsterdamer, and Tova Milo. Uncertainty in crowd data sourcing under structural constraints. In *UnCrowd*, 2014.

5 Antoine Amarilli, Yael Amsterdamer, Tova Milo, and Pierre Senellart. Top-k querying of unknown values under order constraints (extended version). *CoRR*, abs/1701.02634, 2017.

6 Antoine Amarilli, Pierre Bourhis, and Pierre Senellart. Provenance circuits for trees and treelike instances. In *ICALP*, 2015. `doi:10.1007/978-3-662-47666-6_5`.

7 Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *JACM*, 51(4), 2004.

8 Christopher M. Bishop. Graphical models. In *Pattern Recognition and Machine Learning*, chapter 8. Springer, 2006.

9 Jonathan Bragg, Mausam, and Daniel S. Weld. Crowdsourcing multi-label classification for taxonomy creation. In *HCOMP*, 2013.

10 Graham Brightwell and Peter Winkler. Counting linear extensions. *Order*, 8(3), 1991.

11 Benno Büeler, Andreas Enge, and Komei Fukuda. Exact volume computation for polytopes: a practical study. In *Polytopes – combinatorics and Computation*, 2000.

12 Reynold Cheng, Dmitri V Kalashnikov, and Sunil Prabhakar. Evaluating probabilistic queries over imprecise data. In *SIGMOD*, 2003.

13 Eleonora Ciceri, Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. Crowdsourcing for top-k query processing over uncertain data. *IEEE TKDE*, 28(1), 2016.

14 Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. Running tree automata on probabilistic XML. In *PODS*, 2009.

15 Graham Cormode, Feifei Li, and Ke Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, 2009.

16 Ben Cousins and Santosh Vempala. A practical volume algorithm. *Mathematical Programming Computation*, 8(2), 2016.

17 Susan B. Davidson, Sanjeev Khanna, Tova Milo, and Sudeepa Roy. Using the crowd for top-k and group-by queries. In *ICDT*, 2013.

18 Jesús A De Loera, B Dutra, Matthias Köppe, S Moreinis, G Pinto, and J Wu. Software for exact integration of polynomials over polyhedra. *Computational Geometry*, 46(3), 2013.

19 Landon Detwiler, Wolfgang Gatterbauer, Brenton Louie, Dan Suciu, and Peter Tarczy-Hornoch. Integrating and ranking uncertain scientific data. In *ICDE*, 2009.

20 Ulrich Faigle, Laszlo Lovasz, Rainer Schrader, and Gy Turán. Searching in trees, series-parallel and interval orders. *SIAM J. Comput.*, 15(4), 1986.

21 Cunjing Ge and Feifei Ma. A fast and practical method to estimate volumes of convex polytopes. In *FAW*, 2015.

22 James E. Gentle. *Computational Statistics*. Springer, 2009.

23 Google Product Taxonomy, 2016. `https://support.google.com/merchants/answer/1705911?hl=en`.

24 Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharma. Discovering all most specific sentences. *TODS*, 28(2), 2003.

25 Parisa Haghani, Sebastian Michel, and Karl Aberer. Evaluating top-k queries over incomplete data streams. In *CIKM*, 2009.

26 Ming Hua, Jian Pei, and Xuemin Lin. Ranking queries on uncertain data. *VLDB J.*, 20(1), 2011.

27 Yannis E. Ioannidis and Raghu Ramakrishnan. Efficient transitive closure algorithms. In *VLDB*, 1988.

28 Jeffrey Jestes, Graham Cormode, Feifei Li, and Ke Yi. Semantics of ranking queries for probabilistic data. *IEEE TKDE*, 23(12), 2011.

29 Frank Jones. *Lebesgue Integration on Euclidean Space*. Jones & Bartlett Learning, 2001.

30 Kustaa Kangas, Teemu Hankala, Teppo Niinimäki, and Mikko Koivisto. Counting linear extensions of sparse posets. In *IJCAI*, 2016.

31 Ravi Kannan, László Lovász, and Miklós Simonovits. Random walks and an $O^*(n^5)$ volume algorithm for convex bodies. *Random Struct. Algorithms*, 11(1), 1997.

32 Jim Lawrence. Polytope volume computation. *Mathematics of Computation*, 57(195), 1991.

33 Jian Li, Barna Saha, and Amol Deshpande. A unified approach to ranking in probabilistic databases. *PVLDB*, 2(1), 2009.

34 Xiang Lian and Lei Chen. Probabilistic ranked queries in uncertain databases. In *EDBT*, 2008.

35 Xiang Lian and Lei Chen. A generic framework for handling uncertain data with local correlations. *VLDB*, 4(1), 2010.

36 László Lovász and István Deák. Computational results of an $\mathrm{O}^*(n^4)$ volume algorithm. *European J. Operational Research*, 216(1), 2012.

37 László Lovász and Santosh Vempala. Hit-and-run from a corner. *SIAM J. Comput.*, 35(4), 2006.

38 Frederic Maire. An algorithm for the exact computation of the centroid of higher dimensional polyhedra and its application to kernel machines. In *ICDM*, 2003.

39 Christos H Papadimitriou. Efficient search for rationals. *Information Processing Letters*, 8(1), 1979.

40 A. Parameswaran, A.D. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *PVLDB*, 4(5), 2011.

41 Gara Pruesse and Frank Ruskey. Generating linear extensions fast. *SIAM J. Comput.*, 23(2), 1994.

42 Luis A Rademacher. Approximating the centroid is hard. In *SCG*, 2007.

43 Christopher Re, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.

44 Alexander Schrijver. The structure of polyhedra. In *Theory of Linear and Integer Programming*, chapter 8. Wiley-Interscience, 1986.

45 Miklós Simonovits. How to compute the volume in high dimension? *Mathematical programming*, 97(1-2), 2003.

46 Mohamed A. Soliman, Ihab F. Ilyas, and Shalev Ben-David. Supporting ranking queries on uncertain and incomplete data. *VLDB J.*, 19(4), 2010.

47 Mohamed A. Soliman, Ihab F. Ilyas, and K. Chen-Chuan Chang. Top-k query processing in uncertain databases. In *ICDE*, 2007.

48 Julia Stoyanovich, Sihem Amer-Yahia, Susan B Davidson, Marie Jacob, Tova Milo, et al. Understanding local structure in ranked datasets. In *CIDR*, 2013.

49 Chong Sun, Narasimhan Rampalli, Frank Yang, and AnHai Doan. Chimera: Large-scale classification using machine learning, rules, and crowdsourcing. *PVLDB*, 7(13), 2014.

50 Chonghai Wang, Li-Yan Yuan, Jia-Huai You, Osmar R. Zaïane, and Jian Pei. On pruning for top-k ranking in uncertain databases. *PVLDB*, 4(10), 2011.

51 Ke Yi, Feifei Li, George Kollios, and Divesh Srivastava. Efficient processing of top-k queries in uncertain databases. In *ICDE*, 2008.

52 Xi Zhang and Jan Chomicki. Semantics and evaluation of top-k queries in probabilistic databases. *DAPD*, 26(1), 2009.