On the Semantics and Evaluation of Top-k Queries in Probabilistic Databases

Xi Zhang, Jan Chomicki

Department of Computer Science and Engineering,
University at Buffalo, SUNY, U.S.A.
{xizhang,chomicki}@cse.buffalo.edu

***Abstract*— We formulate three intuitive semantic properties for top-k queries in probabilistic databases, and propose Global-Top$k$ query semantics which satisfies all of them. We provide a dynamic programming algorithm to evaluate top-k queries under Global-Top$k$ semantics in simple probabilistic relations. For general probabilistic relations, we show a polynomial reduction to the simple case. Our analysis shows that the complexity of query evaluation is linear in k and at most quadratic in database size.**

I. Introduction

The study of incompleteness and uncertainty in databases has long been an interest of the database community [1], [2], [3], [4], [5], [6], [7]. Recently, this interest has been rekindled by an increasing demand for managing rich data, often incomplete and uncertain, emerging from scientific data management, sensor data management, data cleaning, information extraction etc. [8] focuses on query evaluation in traditional probabilistic databases; ULDB [9] supports uncertain data and data lineage in Trio [10]; MayBMS [11] uses the vertical World-Set representation of uncertain data [12]. The standard semantics adopted in most works is the *possible worlds* semantics [1], [5], [6], [9], [8], [12].

On the other hand, since the seminal papers of Fagin [13], [14], the top-k problem has been extensively studied in multimedia databases [15], middleware systems [16], data cleaning [17], core technology in relational databases [18], [19] etc. In the top-k problem, each tuple is given a *score*, and users are interested in the k tuples with the highest score. More recently, the top-k problem has been studied in probabilistic databases [20], [21]. Those papers, however, are solving two essentially different top-k problems. Soliman et al. [20] assumes the existence of a scoring function to rank tuples. Probabilities provide information on how likely tuples will appear in the database. In contrast, in [21], the ranking criterion for top-k is the probability associated with each query answer. In many applications, it is necessary to deal with tuple probabilities and scores at the same time. Thus, in this paper, we use the model of [20]. Even in this model, different semantics for top-k queries are possible, so a part of the challenge is to define a reasonable semantics.

As a motivating example, the following smart environment scenario is inspired by the work in [22].

Example 1: A smart lab has the following data from a Saturday night:

Name	Biometric Score (Face, Voice, ...)	Prob. of Sat Nights
Aidan	0.65	0.3
Bob	0.55	0.9
Chris	0.45	0.4

Typically, the lab collects two kinds of data: biometric data from sensors and historical statistics. Biometric data comes from the sensors deployed in the lab, for example, face recognition and voice recognition sensors. This data is collected and matched against the profile of each person involved in the lab. It can be further normalized to give us an idea of how well each person fits the sensed data. In addition, the lab also keeps track of the statistics of each person's activities.

Knowing that we definitely had two visitors that night, we would like to ask the following question: "Who were the two visitors in the lab last Saturday night?" This question can be formulated as a top-k query over the above probabilistic relation, where $k = 2$.

In Example 1, each tuple is associated with an *event*, which is that candidate being in the lab on Saturday nights. The probability of the event is shown next to each tuple. In this example, all the events of tuples are independent, and tuples are therefore said to be *independent*. Intuitively, for the top-k problem in Example 1, we are not necessarily interested in candidates with high biometric scores if the associated events are very unlikely to happen, e.g. we have strong evidence suggesting that a candidate plays football on Saturday nights and his probability of being in lab is 0.001.

Example 1 shows a *simple* probabilistic relation where the tuples are independent. In contrast, Example 2 illustrates a more general case.

Example 2: In a sensor network deployed in a habitat, each sensor reading comes with a confidence value *Prob*, which is the probability that the reading is valid. The following table shows the temperature sensor readings at a given sampling time. These data are from two sensors, Sensor 1 and Sensor 2, which correspond to two *parts* of the relation, marked C_1 and C_2 respectively. Each sensor has only one *true* reading at a given time, therefore tuples from the same part of the relation correspond to exclusive events.

	Temp.(°F)	Prob
C_1	22	0.6
	10	0.4
C_2	25	0.1
	15	0.6

Our question is:

"What's the temperature of the warmest spot?"

The question can be formulated as a top-k query, where $k = 1$, over a probabilistic relation containing the above data. However, we must take into consideration that the tuples in each part $C_i, i = 1, 2$, are exclusive.

Our contributions in this paper are the following:

- We formulate three intuitive semantic properties and use them to compare different top-k semantics in probabilistic databases (Section III-A);
- We propose a new semantics for top-k queries in probabilistic databases, called Global-Top$k$, which satisfies all the properties above (Section III-B);
- We exhibit efficient algorithms for evaluating top-k queries under the Global-Top$k$ semantics in simple probabilistic databases (Section IV-A) and general probabilistic databases (Section IV-C).

II. Background

A. Probabilistic Relations

To simplify the discussion in this paper, a probabilistic database contains a single *probabilistic relation*. We refer to a traditional database relation as a *deterministic relation*. A deterministic relation R is a set of tuples. A *partition* $\mathcal{C}$ of R is a collection of non-empty subsets of R such that every tuple belongs to one and only one of the subsets. That is, $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ such that $C_1 \cup C_2 \cup \ldots \cup C_m = R$ and $C_i \cap C_j = \emptyset, 1 \leq i \neq j \leq m$. Each subset $C_i, i = 1, 2, \ldots, m$ is a *part* of the partition $\mathcal{C}$. A *probabilistic relation* R^p has three components, a *support (deterministic) relation* R, a probability function p and a partition $\mathcal{C}$ of the support relation R. The probability function p maps every tuple in R to a probability value in $(0, 1]$. The partition $\mathcal{C}$ divides R into subsets such that the tuples within each subset are exclusive and therefore their probabilities sum up to at most 1. In the graphical presentation of R, we use horizontal lines to separate tuples from different parts.

Definition 2.1 (Probabilistic Relation): A probabilistic relation R^p is a triplet $\langle R, p, \mathcal{C}\rangle$, where R is a support deterministic relation, p is a probability function $p : R \mapsto (0, 1]$ and $\mathcal{C}$ is a partition of R such that $\forall C_i \in \mathcal{C}, \sum_{t \in C_i} p(t) \leq 1$.

In addition, we make the assumption that tuples from different parts of of $\mathcal{C}$ are independent, and tuples within the same part are exclusive. Def. 2.1 is equivalent to the model used in Soliman et al. [20] with exclusive tuple generation rules. Ré et al. [21] proposes a more general model, however only a restricted model equivalent to Def. 2.1 is used in top-k query evaluation.

Example 2 shows an example of a probabilistic relation whose partition has two parts. Generally, each part corresponds to a real world entity, in this case, a sensor. Since there is only one true state of an entity, tuples from the same part are exclusive. Moreover, the probabilities of all possible states of an entity sum up to at most 1. In Example 2, the sum of probabilities of the tuples from Sensor 1 is 1, while that from Sensor 2 is 0.7. This can happen for various reasons. In the above example, we might encounter a physical difficulty in collecting the sensor data, and end up with partial data.

Definition 2.2 (Simple Probabilistic Relation): A probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ is simple iff the partition $\mathcal{C}$ contains only singleton sets.

The probablistic relation in Example 1 is simple (individual parts not illustrated). Note that in this case, $|R| = |\mathcal{C}|$.

We adopt the well-known *possible worlds* semantics for probabilistic relations [1], [5], [6], [9], [8], [12].

Definition 2.3 (Possible World): Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a deterministic relation W is a *possible world* of R^p iff

1) W is a subset of the support relation, i.e. $W \subseteq R$;
2) For every part C_i in the partition $\mathcal{C}$, at most one tuple from C_i is in W, i.e. $\forall C_i \in \mathcal{C}, |C_i \cap W| \leq 1$.

Denote by $pwd(R^p)$ *the set of all possible worlds* of R^p. Since all the parts in $\mathcal{C}$ are independent, we have the following definition of the probability of a possible world.

Definition 2.4 (Probability of a Possible World): Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, for any $W \in pwd(R^p)$, its probability $Pr(W)$ is defined as

$$Pr(W) = \prod_{t \in W} p(t) \prod_{C_i \in \mathcal{C}'} (1 - \sum_{t \in C_i} p(t)) \qquad (1)$$

where $\mathcal{C}' = \{C_i \in \mathcal{C} | W \cap C_i = \emptyset\}$.

B. Scoring Functions

A *scoring function over a deterministic relation* R is a function from R to real numbers, i.e. $s : R \mapsto \mathbb{R}$. The function s induces a *preference relation* $\succ_s$ and an *indifference relation* $\sim_s$ on R. For any two distinct tuples t_i and t_j from R,

$$t_i \succ_s t_j \text{ iff } s(t_i) > s(t_j);$$
$$t_i \sim_s t_j \text{ iff } s(t_i) = s(t_j).$$

When the scoring function s is *injective*, $\succ_s$ establishes a *total order*, i.e. an irreflective, transitive, connected binary relation, over R. In such a case, no two tuples from R tie in score.

A *scoring function over a probabilistic relation* $R^p = \langle R, p, \mathcal{C}\rangle$ is a scoring function s over its support relation R.

C. Top-k Queries

Definition 2.5: (Top-k Answer Set over Deterministic Relation): Given a deterministic relation R, a non-negative integer k and a scoring function s over R, a top-k answer in R under s is a set T of tuples such that

1) $T \subseteq R$;
2) If $|R| < k$, $T = R$, otherwise $|T| = k$;
3) $\forall t \in T, \forall t' \in R - T, t \succ_s t'$ or $t \sim_s t'$.

According to Def. 2.5, given k and s, there can be more than one top-k answer set in a deterministic relation R. The evaluation of a top-k query over R returns one of them nondeterministically, say S. However, if the scoring function s is injective, S is unique, denoted by $top_{k,s}(R)$.

III. Semantics of Top-k Queries

A. Semantic Properties of Top-k Answers

Probability opens the gates for various possible semantics for top-k queries. As the semantics of a probabilistic relation involves a set of worlds, it is to be expected that there may be more than one top-k answer, even under an injective scoring function. The answer to a top-k query over a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ should clearly be a set of tuples from its support relation R. In order to compare different semantics, we formulate below some *properties* we would like any reasonable semantics to satisfy.

In the following discussion, S is any top-k answer set in $R^p = \langle R, p, \mathcal{C}\rangle$ under an injective scoring function s. A tuple from the support relation R is a *winner* if it belongs to some top-k answer set under that semantics, and a *loser* otherwise. That is to say, in the case of multiple top-k answer sets, any tuple from any of them is a winner.

Properties

1) *Exact k*: When R^p is sufficiently large ($|\mathcal{C}| \geq k$), the cardinality of S is exactly k;
2) *Faithfulness*: For any two tuples $t_1, t_2 \in R$, if both the score and the probability of t_1 are higher than those of t_2 and $t_2 \in S$, then $t_1 \in S$;
3) *Stability*:
   - Raising the score/probability of a winner will not turn it into a loser;
   - Lowering the score/probability of a loser will not turn it into a winner.

All of those properties reflect basic intuitions about top-k answers. *Exact k* expresses user expectations about the size of the result. *Faithfulness* and *Stability* reflect the significance of score and probability.

B. Global-Topk Semantics

We propose here a new top-k answer semantics in probabilistic relations, namely *Global-Topk*, which satisfies all the properties formulated in Section III-A:

- **Global-Top**k: return k highest-ranked tuples according to their probability of being in the top-k answers in possible worlds.

Considering a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ under an injective scoring function s, any $W \in pwd(R^p)$ has a unique top-k answer set $top_{k,s}(W)$. Each tuple from the support relation R can be in the top-k answer (in the sense of Def. 2.5) in zero, one or more possible worlds of R^p. Therefore, the sum of the probabilities of those possible worlds provides a global ranking criterion.

Definition 3.1 (Global-Topk Probability): Assume a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a non-negative integer k and an injective scoring function s over R^p. For any tuple t in R, the Global-Top$k$ probability of t, denoted by $P^{R^p}_{k,s}(t)$, is the sum of the probabilities of all possible worlds of R^p whose top-k answer contains t.

$$P^{R^p}_{k,s}(t) = \sum_{\substack{W \in pwd(R^p) \\ t \in top_{k,s}(W)}} Pr(W).$$

For simplicity, we skip the superscript in $P^{R^p}_{k,s}(t)$, i.e. $P_{k,s}(t)$, when the context is unambiguous.

Definition 3.2: (Global-Topk Answer Set over Probabilistic Relation): Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a non-negative integer k and an injective scoring function s over R^p, a top-k answer in R^p under s is a set T of tuples such that

1) $T \subseteq R$;
2) If $|R| < k$, $T = R$, otherwise $|T| = k$;
3) $\forall t \in T, \forall t' \in R - T, P_{k,s}(t) \geq P_{k,s}(t')$.

Notice the similarity between Def. 3.2 and Def. 2.5. In fact, the probabilistic version only changes the last condition, which restates the preferred relationship between two tuples by taking probability into account. This semantics preserves the nondeterministic nature of Def. 2.5. For example, if two tuples are of the same Global-Top$k$ probability, and there are $k-1$ tuples with higher Global-Top$k$ probability, Def. 2.5 allows one of the two tuples to be added to the top-k answer nondeterministically. Example 3 gives an example of the Global-Top$k$ semantics.

Example 3: Consider the top-2 query in Example 1. Clearly, the scoring function here is the biometric scoring function. The following table shows all the possible worlds and their probabilities. For each world, the underlined people are in the top-2 answer set of that world.

Possible World	Prob
$W_1 = \emptyset$	0.042
$W_2 = \{\underline{Aidan}\}$	0.018
$W_3 = \{\underline{Bob}\}$	0.378
$W_4 = \{\underline{Chris}\}$	0.028
$W_5 = \{\underline{Aidan}, \underline{Bob}\}$	0.162
$W_6 = \{\underline{Aidan}, \underline{Chris}\}$	0.012
$W_7 = \{\underline{Bob}, \underline{Chris}\}$	0.252
$W_8 = \{\underline{Aidan}, \underline{Bob}, Chris\}$	0.108

Chris is in the top-2 answer of W_4, W_6, W_7, so its top-2 probability is $0.028 + 0.012 + 0.252 = 0.292$. Similarly, the top-2 probability of Aidan and Bob are 0.9 and 0.3 respectively. $0.9 > 0.3 > 0.292$, therefore Global-Top$k$ will return $\{Aidan, Bob\}$.

Note that top-k answer sets may be of cardinality less than k for some possible worlds. We refer to such possible worlds as *small* worlds. In Example 3, $W_{1\ldots4}$ are all small worlds.

C. Other Semantics

Soliman et al. [20] proposes two semantics for top-k queries in probabilistic relations.

- *U-Top$k$*: return the most probable top-k answer set that belongs to possible world(s);

- *U-kRanks*: for $i = 1, 2, \ldots, k$, return the most probable i^{th}-ranked tuples across all possible worlds.

Example 4: Continuing Example 3, under U-Top$k$ semantics, the probability of top-2 answer set $\{Bob\}$ is 0.378, and that of $\{Aidan, Bob\}$ is $0.162 + 0.108 = 0.27$. Therefore, $\{Bob\}$ is more probable than $\{Aidan, Bob\}$ under U-Top$k$. In fact, $\{Bob\}$ is the most probable top-2 answer set in this case, and will be returned by U-Top$k$.

Under U-kRanks semantics, Aidan is in 1^{st} place in the top-2 answer of W_2, W_5, W_6, W_8, therefore the probability of Aidan being in 1^{st} place in the top-2 answers in possible worlds is $0.018 + 0.162 + 0.012 + 0.108 = 0.3$. However, Aidan is not in 2^{nd} place in the top-2 answer of any possible world, therefore the probability of Aidan being in 2^{nd} place is 0. In fact, we can construct the following table.

	Aidan	Bob	Chris
Rank 1	0.3	<u>0.63</u>	0.028
Rank 2	0	<u>0.27</u>	0.264

U-kRanks selects the tuple with the highest probability at each rank (underlined) and takes the union of them. In this example, Bob wins at both Rank 1 and Rank 2. Thus, the top-2 answer returned by U-kRanks is $\{Bob\}$.

The properties introduced in Section III-A lay the ground for comparing different semantics. In Table I, a single "✓" (resp. "×") indicates that property is (resp. is not) satisfied under that semantics. "✓/×" indicates that, the property is satisfied by that semantics in *simple* probabilistic relations, but not in the general case.

TABLE I
PROPERTY SATISFACTION FOR DIFFERENT SEMANTICS

Semantics	Exact k	Faithfulness	Stability
Global-Top$k$	✓	✓	✓
U-Top$k$	×	✓/×	✓
U-kRanks	×	×	×

Global-Top$k$ satisfies all the properties while neither of the other two semantics does. For *Exact k*, Global-Top$k$ is the only one that satisfies this property. Example 4 illustrates the case when both U-Top$k$ and U-kRanks violate this property. It is not satisfied by U-Top$k$ because a *small* possible world with high probability could dominate other worlds. In that case, the dominating possible world might not have enough tuples. It is also violated by U-kRanks because a single tuple can win at multiple ranks in U-kRanks. For *Faithfulness*, since U-Top$k$ requires all tuples in a top-k answer set to be compatible, this property can be violated when a high-score/probability tuple could be dragged down arbitrarily by its compatible tuples if they are not very likely to appear. U-kRanks violates both *Faithfulness* and *Stability*. Under U-kRanks, instead of a set, a top-k answer is an ordered vector, where ranks are significant. A change in a tuple's probability/score might have unpredictable consequence on ranks, therefore those two properties are not guaranteed to hold.

IV. QUERY EVALUATION UNDER GLOBAL-TOPk

A. Simple Probabilistic Relations

We first consider a *simple* probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ under an injective scoring function s.

Theorem 4.1: Given a simple probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ and an injective scoring function s over R^p, if $R = \{t_1, t_2, \ldots, t_n\}$ and $t_1 \succ_s t_2 \succ_s \ldots \succ_s t_n$, the following recursion on Global-Top$k$ queries holds.

$$q(k,i) = \begin{cases} 0 & k = 0 \\ p(t_i) & 1 \le i \le k \\ (q(k, i-1)\frac{\bar{p}(t_{i-1})}{p(t_{i-1})} + q(k-1, i-1))p(t_i) & \text{otherwise} \end{cases} \quad (2)$$

where $q(k,i) = P_{k,s}(t_i)$ and $\bar{p}(t_{i-1}) = 1 - p(t_{i-1})$.

Proof: (Sketch) Recall that $top_{k,s}(W)$ is the unique top-k answer set of a possible world W under s. In the trivial case when $k = 0$, $top_{0,s}(W)$ is an empty set for any $W \in pwd(R^p)$. Therefore, $q(0,i) = 0$ for all $t_i, i = 1, 2, \ldots n$.

For any of the k tuples with the highest score in R, namely $t_1, t_2, \ldots, t_k$, whenever it appears in a possible world W, it would be in $top_{k,s}(W)$. In other words, its Global-Top$k$ probability equals to its tuple probability.

For any other tuple $t_i, i > k$, t_i is in $top_{k,s}(W)$ of a possible world W as long as there are no more than $k-1$ tuples with higher score in W. Moreover, in a simple probabilistic relation, Eqn. (1) is equivalent to the following equation:

$$Pr(W) = \prod_{t \in W} p(t) \prod_{t \in R - W} \bar{p}(t) \quad (3)$$

Assume we know the Global-Top$k$ probability of t_{i-1}, i.e. $q(k, i-1)$, if we substitute t_{i-1} with t_i in every world W contributing to $q(k, i-1)$, t_i will then be in the top-k answer of the new world. Those new worlds contribute to the first part of the third case in Eqn. (2). Assume we know the Global-Top$(k-1)$ probability of t_{i-1}, i.e. $q(k-1, i-1)$, for any world W contributing to $q(k-1, i-1)$, if W contains t_i, t_i will also be in $top_{k,s}(W)$. Those worlds contribute to the second part of the third case in Eqn. (2). ■

Example 5: Consider a simple probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, where $R = \{t_1, t_2, t_3, t_4\}$, $p(t_i) = p_i, 1 \le i \le 4$, $\mathcal{C} = \{\{t_1\}, \{t_2\}, \{t_3\}, \{t_4\}\}$ and an injective scoring function s such that $t_1 \succ_s t_2 \succ_s t_3 \succ_s t_4$. The following table shows the Global-Top$k$ of t_i, where $0 \le k \le 2$.

k	t_1	t_2	t_3	t_4
0	0	0	0	0
1	p_1	$\bar{p}_1 p_2$	$\bar{p}_1\bar{p}_2 p_3$	$\bar{p}_1\bar{p}_2\bar{p}_3 p_4$
2	$\mathbf{p_1}$	$\mathbf{p_2}$	$(\bar{\mathbf{p}}_\mathbf{2} + \bar{\mathbf{p}}_\mathbf{1}\mathbf{p_2})\mathbf{p_3}$	$((\bar{\mathbf{p}}_\mathbf{2} + \bar{\mathbf{p}}_\mathbf{1}\mathbf{p_2})\bar{\mathbf{p}}_\mathbf{3} + \bar{\mathbf{p}}_\mathbf{1}\bar{\mathbf{p}}_\mathbf{2}\mathbf{p_3})\mathbf{p_4}$

Row 2 (bold) is each t_i's Global-Top2 probability. Now, if we are interested in top-2 answer in R^p, we only need to pick the two tuples with the highest value in Row 2.

Given the recursion in Thm. 4.1, we can apply the standard dynamic programming (DP) technique, together with a priority queue, to select k tuples with the highest Global-Top$k$

probability, as shown in Alg. 1. It is a one-pass computation on the probabilistic relation, which can be easily implemented even if secondary storage is used. The overhead is the initial sorting cost (not shown in Alg. 1), which would be amortized by the workload of consecutive top-k queries.

Algorithm 1 (Ind_Topk) Evaluate Global-Top$k$ queries in a Simple Probabilistic Relation

Require: $R^p = \langle R, p, \mathcal{C}\rangle, k$

```
Initialize a fixed cardinality (k+1) priority queue Ans of ⟨t, prob⟩ pairs, which compares pairs on prob;
q(0,1) = 0;
for k' = 1 to k do
    q(k',1) = p(t_{k'});
end for
for i = 2 to |R| do
    for k' = 0 to k do
        if k' = 0 then
            q(k',i) = 0;
        else
            q(k',i) =
            p(t_i)(q(k',i-1) p̄(t_{i-1})/p(t_{i-1}) + q(k'-1,i-1));
        end if
    end for
    Add ⟨t_i, q(k,i)⟩ to Ans;
    if |Ans| > k then
        remove the pair with the smallest prob value from Ans;
    end if
end for
return {t_i | ⟨t_i, q(k,i)⟩ ∈ Ans};
```

B. Threshold Algorithm Optimization

Fagin [14] proposes *Threshold Algorithm(*TA*)* for processing top-k queries in a middleware scenario. In a middleware system, an *object* has m attributes. For each attribute, there is a sorted list ranking objects in the decreasing order of its score on that attribute. An *aggregation function* f combines the individual attribute scores x_i, i=$1, 2, \ldots, m$ to obtain the overall object score $f(x_1, x_2, \ldots, x_m)$. An aggregation function is *monotonic* iff $f(x_1, x_2, \ldots, x_m) \leq f(x'_1, x'_2, \ldots, x'_m)$ whenever $x_i \leq x'_i$ for every i. Fagin [14] shows that TA is cost-optimal in finding the top-k objects in such a system.

TA is guaranteed to work as long as the aggregation function is monotonic. For a simple probabilistic relation, if we regard *score* and *probability* as two special attributes, Global-Top$k$ probability $P_{k,s}$ is an aggregation function of *score* and *probability*. The *Faithfulness* property in Section III-A implies the monotonicity of Global-Top$k$ probability. Consequently, assuming that we have an index on probability as well, we can guide the dynamic programming(DP) in Alg. 1 by TA. Now, instead of computing all kn entries for DP, where $n = |R|$, the algorithm can be stopped as early as possible. A subtlety is that Global-Top$k$ probability $P_{k,s}$ is *only* well-defined for $t \in R$, unlike in [14], where an aggregation function is well-defined over the domain of all possible attribute values. Therefore, compared to the original TA, we need to achieve the same behavior without referring to virtual tuples which are not in R.

U-Top$k$ satisfies *Faithfulness* in simple probabilistic relations, but the candidates under U-Top$k$ are *sets* not *tuples* and thus there is no counterpart of an aggregation function under U-Top$k$. Therefore, TA is not applicable. Neither is TA applicable to U-kRanks. Though we can define an aggregation function per $rank$, $rank = 1, 2, \ldots, k$, for tuples under U-kRanks, the violation of *Faithfulness* in Table I suggests a violation of monotonicity of those k aggregation functions.

Denote T and P for the list of tuples in the decreasing order of score and probability respectively. Following the convention in [14], $\underline{t}$ and $\underline{p}$ are the last value seen in T and P respectively.

Applying TA to Global-Topk Computation.

(1) Go down T list, and fill in entries in the DP table. Specifically, for $\underline{t} = t_j$, compute the entries in the j^{th} column up to the k^{th} row. Add t_j to the top-k answer set Ans, if any of the following conditions holds:
   (a) Ans has less than k tuples, i.e. $|Ans| < k$;
   (b) The Global-Top$k$ probability of t_j, i.e. $q(k, j)$, is greater than the lower bound of Ans, i.e. LB_{Ans}, where $LB_{Ans} = \min_{t_i \in Ans} q(k, i)$.

   In the second case, we also need to drop the tuple with the lowest Global-Top$k$ probability in order to keep the cardinality of Ans.

(2) After we have seen at least k tuples in T, we go down P list to find the first p whose tuple t has not been seen. Let $\underline{p} = p$, and we can use $\underline{p}$ to estimate the *threshold*, i.e. upper bound (UP) of the Global-Top$k$ probability of any unseen tuple. Assume $\underline{t} = t_i$,

$$UP = (q(k, i)\frac{\bar{p}(t_i)}{p(t_i)} + q(k-1, i))\underline{p}.$$

(3) If $UP > LB_{Ans}$, we can expect Ans will be updated in the future, so go back to (1). Otherwise, we can stop safely and report Ans.

Theorem 4.2 (Correctness): Given a simple probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, a non-negative integer k and an injective scoring function s over R^p, the above TA-based algorithm correctly finds a top-k answer under Global-Top$k$ semantics.

Proof: In every iteration of Step (2), say $\underline{t} = t_i$, for any unseen tuple t, s' is an injective scoring function over R^p, which only differs from s in the score of t. Under the function s', $t_i \succ_{s'} t \succ_{s'} t_{i+1}$. If we evaluate the top-k query in R^p under s' instead of s, $P_{k,s'}(t) = \frac{p(t)}{\underline{p}}UP$. On the other hand, for any $W \in pwd(R^p)$, W contributing to $P_{k,s}(t)$ implies that W contributes to $P_{k,s'}(t)$, while the reverse is not necessarily true. So, we have $P_{k,s'}(t) \geq P_{k,s}(t)$. Recall that $\underline{p} \geq p(t)$, therefore $UP \geq \frac{p(t)}{\underline{p}}UP = P_{k,s'}(t) \geq P_{k,s}(t)$. The conclusion follows from the correctness of the original TA algorithm and Alg. 1. ■

The optimization above aims at an early stop. Bruno et al. [23] carries out an extensive experimental study on the effectiveness of applying TA in RDBMS. They consider various aspects of query processing. Note that *probability* is typically supported as a special attribute in DBMS. One of their conclusions is that if at least one of the indices available for the attributes is a *covering index*, that is, it is defined over all other attributes and we can get the values of all other attributes directly without performing a primary index lookup, then the improvement by TA can be up to two orders of magnitude. The cost of building a useful set of indices once would be amortized by a large number of top-k queries that subsequently benefit form such indices. Even in the lack of covering indices, if the data is highly correlated, in our case, that means high-score tuples having high probabilities, TA would still be effective.

C. Arbitrary Probabilistic Relations

1) Induced Event Relation: In the general case of probabilistic relation, each part of of the partition $\mathcal{C}$ can contain more than one tuple. The crucial *independence* assumption in Alg. 1 no longer holds. However, even though tuples are not independent, *parts* of the partition $\mathcal{C}$ are. In the following definition , we assume an identifier function id. For any tuple t, $id(t)$ is the identifier of the part where t belongs.

Definition 4.1 (Induced Event Relation): Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$, an injective scoring function s over R^p and a tuple $t \in C_{id(t)} \in \mathcal{C}$, the event relation induced by t, denoted by $E^p = \langle E, p^E, \mathcal{C}^E\rangle$, is a probabilistic relation whose support relation E has only one attribute, $Event$. E and the probability function p^E are defined by the following two generation rules:

- Rule 1:
$$t_{e_t} \in E \text{ and } p^E(t_{e_t}) = p(t);$$
- Rule 2:
$$\forall C_i \in \mathcal{C} \wedge C_i \neq C_{id(t)}, [(\exists t' \in C_i \wedge t' \succ_s t) \Rightarrow (t_{e_{C_i}} \in E) \text{ and } p^E(t_{e_{C_i}}) = \sum_{t' \succ_s t} p(t')].$$

No other tuples belong to E. The partition $\mathcal{C}^E$ is defined as the collection of singleton subsets of E.

Except for one special tuple generated by *Rule 1*, each tuple in the induced event relation (generated by *Rule 2*) represents an event e_{C_i} associated with a part $C_i \in \mathcal{C}$. The probability of this event, denoted by $p(t_{e_{C_i}})$, is the probability that e_{C_i} occurs. Given the tuple t, the *event* e_{C_i} is defined as "some tuple from the part C_i has the score higher than the score of t".

The role of the special tuple t_{e_t} and its probability $p(t)$ will become clear in Thm. 4.3. Let us first look at an example of an induced event relation.

Example 6: Given R^p as in Example 2, we would like to construct the induced event relation $E^p = \langle E, p^E, \mathcal{C}^E\rangle$ for tuple t=(Temp: 15) from C_2. By Rule 1, we have $t_{e_t} \in E$, $p^E(t_{e_t}) = 0.6$. By Rule 2, since $t \in C_2$, we have $t_{e_{C_1}} \in E$ and $p^E(t_{e_{C_1}}) = \sum_{t' \succ_s t} p(t') = p((\text{Temp: } 22)) = 0.6$. Therefore,

E: p^E:

Event	Prob
t_{e_t}	0.6
$t_{e_{C_1}}$	0.6

Proposition 4.1: Any induced event relation is a simple probabilistic relation.

2) Evaluating Global-Topk Queries: With the help of *induced event relation*, we could reduce Global-Top$k$ in the general case to Global-Top$k$ in simple probabilistic relations.

Lemma 4.1: Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ and an injective scoring function s, for any $t \in R$, $E^p = \langle E, p^E, \mathcal{C}^E\rangle$. Let $Q^p = \langle E - \{t_{e_t}\}, p^E, \mathcal{C}^E - \{\{t_{e_t}\}\}\rangle$. Then, the Global-Top$k$ probability of t satisfies the following:

$$P^{R^p}_{k,s}(t) = p(t) \cdot \Big(\sum_{\substack{W_e \in pwd(Q^p) \\ |W_e| < k}} p(W_e) \Big).$$

Theorem 4.3: Given a probabilistic relation $R^p = \langle R, p, \mathcal{C}\rangle$ and an injective scoring function s, for any $t \in R^p$, the Global-Top$k$ probability of t equals the Global-Top$k$ probability of t_{e_t} when evaluating top-k in the induced event relation $E^p = \langle E, p^E, \mathcal{C}^E\rangle$ under the injective scoring function $s^E : E \to \mathbb{R}$, $s^E(t_{e_t}) = \frac{1}{2}$ and $s^E(t_{e_{C_i}}) = i$:

$$P^{R^p}_{k,s}(t) = P^{E^p}_{k,s^E}(t_{e_t}).$$

Proof: (Sketch) When evaluating top-k in E^p under s^E, t_{e_t} is the $|E|^{\text{th}}$ tuple. $P^{E^p}_{k,s}(t_{e_t})$ is the probability of t_{e_t} being present in top-k answers in the possible worlds of E^p. Since t_{e_t} has the lowest score, for any world W contributing to $P^{E^p}_{k,s}(t_{e_t})$, W contains t_{e_t} and has at most $k-1$ tuples besides t_{e_t}. There is a bijection from such W to the W_e in Lemma 4.1. The conclusion follows from Lemma 4.1. ■

Since any induced event relation is simple (Prop. 4.1), Thm. 4.3 illustrates how we can reduce the computation of $P^{R^p}_{k,s}(t)$ in the original probabilistic relation to a top-k computation in a simple probabilistic relation, where we can apply the DP technique in Section IV-A. The complete algorithms are shown below.

Algorithm 2 (IndEx_Topk) Evaluate Global-Top$k$ queries in a General Probabilistic Relation

Require: $R^p = \langle R, p, \mathcal{C}\rangle, k, s$

1: Initialize a fixed cardinality $k+1$ priority queue Ans of $\langle t, prob\rangle$ pairs, which compares pairs on $prob$;
2: **for** $t \in R$ **do**
3: Calculate $P^{R^p}_{k,s}(t)$ using Algorithm 3
4: Add $\langle t, P^{R^p}_{k,s}(t)\rangle$ to Ans;
5: **if** $|Ans| > k$ **then**
6: remove the pair with the smallest $prob$ value from Ans;
7: **end if**
8: **end for**
9: **return** $\{t | \langle t, P^{R^p}_{k,s}(t)\rangle \in Ans\}$;

In Alg. 3, we first find the part $C_{id(t)}$ where t belongs. In Line 2, we initialize the support relation E of the induced

event relation by the tuple generated by Rule 1 in Def. 4.1. For any part C_i other than $C_{id(t)}$(Line 3-8), we compute the probability of the event e_{C_i} according to Def. 4.1. Since all the tuples from the same part are exclusive, this probability is the sum of the probabilities of all tuples that qualify in that part. Note that if no tuple from C_i qualifies, this probability is zero. In this case, we do not care whether any tuple from C_i will be in the possible world or not, since it does not have any influence on whether t will be in top-k or not. The corresponding event tuple is therefore excluded from E. By default, any probabilistic database assumes that any tuple not in the support relation is with probability zero. Line 9 uses Alg. 1 to compute $P_{k,s}^{E^p}(t_{e_t})$. Consequently, we retain only the DP related codes in Alg. 1. Note that Alg. 1 requires all tuples be sorted on score, but this is not a problem for us. Since we already know the scoring function s^E, we simply need to organize tuples based on s^E when generating E. No extra sorting is necessary.

Algorithm 3 (IndEx_Topk_Sub) Calculate $P_{k,s}^{R^p}(t)$using induced event relation

Require: $R^p = \langle R, p, \mathcal{C}\rangle, k, s, t \in R$

1: Find the part $C_{id(t)} \in \mathcal{C}$ such that $t \in C_{id(t)}$;
2: Initialize E with tuple t_{e_t}, where $p^E(t_{e_t}) = p(t)$

$$E = \{t_{e_t}\};$$

3: **for** $C_i \in \mathcal{C}$ and $C_i \neq C_{id(t)}$ **do**
4:

$$p(e_{C_i}) = \sum_{\substack{t' \in C_i \\ t' \succ_s t}} p(t');$$

5: **if** $p(e_{C_i}) > 0$ **then**
6: Add a tuple $t_{e_{C_i}}$ to E, where $p^E(t_{e_{C_i}}) = p(e_{C_i})$

$$E = E \cup \{t_{e_{C_i}}\};$$

7: **end if**
8: **end for**
9: Use Line $2-13$ of Algorithm 1 to compute $P_{k,s^E}^{E^p}(t_{e_t})$;
10: $P_{k,s}^{R^p}(t) = P_{k,s^E}^{E^p}(t_{e_t})$;
11: **return** $P_{k,s}^{R^p}(t)$;

Alg. 2 uses Alg. 3 as a subroutine and computes $P_{k,s}^{R^p}(t)$ for every tuple $t \in R$, then again uses a priority queue to select the final answer set.

D. Complexity

For simple probabilistic relations, in Alg. 1, the DP computation takes $O(kn)$ time and using a priority queue to maintain the k highest values takes $O(nlogk)$. So altogether, Alg. 1 takes $O(kn)$. The TA optimization will reduce the computation time on average, however the algorithm will still have the same complexity.

For general probabilistic relations, in Alg. 3, Line 3-8 takes $O(n)$ to build E (we need to scan all tuples within each part). Line 9 uses DP in Alg. 1, which takes $O(|E|k)$, where $|E|$ is no more than the number of parts in partition $\mathcal{C}$, which is in turn no more than n. So Alg. 3 takes $O(kn)$. Alg. 2 repeats Alg. 3 n times, and the priority queue again takes $O(n \log k)$. Altogether, the complexity is $O(kn^2 + n \log k) = O(kn^2)$.

In [20], both U-Top$k$ and U-kRanks take $O(kn)$ in simple probabilistic relations, which is the same as Alg. 1. In the general case, U-Top$k$ takes $\Theta(kmn^{k-1} \log n)$ and U-kRanks takes $\Omega(mn^{k-1})$, where m is a *rule engine* factor. Both U-Top$k$ and U-kRanks do not scale well with k, for the time complexity is already at least cubic when $k \geq 4$. A detailed analysis is available in [24].

V. Conclusion

We study the semantic and computational problems for top-k queries in probabilistic databases. We propose three desired properties for a top-k semantics, namely *Exact k*, *Faithfulness* and *Stability*. Our Global-Top$k$ semantics satisfies all of them. We study the computational problem of query evaluation under Global-Top$k$ semantics for simple and general probabilistic relations. For the former case, we propose a dynamic programming algorithm and effectively optimize it with Threshold Algorithm. In the latter case, we show a polynomial reduction to the simple case. In contrast to Soliman et al. [20], our approach satisfies intuitive semantic properties and can be implemented more efficiently. However, [20] is of a more general model as it allows arbitrary tuple generation rules.

VI. Future Work

We note that the two dimensions of top-k queries in probabilistic databases, *score* and *probability*, are not treated equally: score is considered in an ordinal sense while probability is considered in a cardinal sense. One of the future directions would be to integrate *strength of preference* expressed by score into our framework. Another direction is to consider *non-injective* scoring functions. A preliminary study shows that this case is non-trivial, because it is not clear how to allocate the probability of a single possible world to different top-k answer sets. Other possible directions include top-k evaluation in other uncertain database models proposed in the literature [12] and more general preference queries in probabilistic databases.

Acknowledgment

Research supported by NSF grant IIS-0307434.

References

[1] T. Imielinski and W. Lipski, "Incomplete information in relational databases," *J. ACM*, vol. 31, no. 4, pp. 761–791, 1984.
[2] R. Cavallo and M. Pittarelli, "The theory of probabilistic databases," in *VLDB*, 1987.
[3] J. Y. Halpern, "An analysis of first-order logics of probability," *Artif. Intell.*, vol. 46, no. 3, pp. 311–350, 1990.
[4] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases : The Logical Level*. Addison Wesley, 1994.
[5] N. Fuhr and T. Rölleke, "A probabilistic relational algebra for the integration of information retrieval and database systems," *ACM Trans. Inf. Syst.*, vol. 15, no. 1, pp. 32–66, 1997.
[6] E. Zimányi, "Query evaluation in probabilistic relational databases," *Theor. Comput. Sci.*, vol. 171, no. 1-2, pp. 179–219, 1997.

[7] L. V. S. Lakshmanan, N. Leone, R. B. Ross, and V. S. Subrahmanian, "Probview: A flexible probabilistic database system," *ACM Trans. Database Syst.*, vol. 22, no. 3, pp. 419–469, 1997.

[8] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," *VLDB J.*, vol. 16, no. 4, pp. 523–544, 2007.

[9] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom, "Uldbs: Databases with uncertainty and lineage," in *VLDB*, 2006.

[10] J. Widom, "Trio: A system for integrated management of data, accuracy, and lineage," in *CIDR*, 2005.

[11] http://www.infosys.uni-sb.de/projects/maybms/.

[12] L. Antova, C. Koch, and D. Olteanu, "World-set decompositions: Expressiveness and efficient algorithms," in *ICDT*, 2007.

[13] R. Fagin, "Combining fuzzy information from multiple systems," *J. Comput. Syst. Sci.*, vol. 58, no. 1, pp. 83–99, 1999.

[14] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware." in *PODS*, 2001.

[15] A. Natsev, Y.-C. Chang, J. R. Smith, C.-S. Li, and J. S. Vitter, "Supporting incremental join queries on ranked inputs," in *VLDB*, 2001.

[16] A. Marian, N. Bruno, and L. Gravano, "Evaluating top- queries over web-accessible databases," *ACM Trans. Database Syst.*, vol. 29, no. 2, pp. 319–362, 2004.

[17] S. Guha, N. Koudas, A. Marathe, and D. Srivastava, "Merging the results of approximate match operations," in *VLDB*, 2004, pp. 636–647.

[18] I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid, "Joining ranked inputs in practice," in *VLDB*, 2002.

[19] ——, "Supporting top-k join queries in relational databases," in *VLDB*, 2003.

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

[21] C. Ré, N. N. Dalvi, and D. Suciu, "Efficient top-k query evaluation on probabilistic data," in *ICDE*, 2007.

[22] V. Menon, B. Jayaraman, and V. Govindaraju, "Biometrics-driven smart environments: Abstract framework and evaluation," CSE Dept., University at Buffalo, SUNY, Tech. Rep. 2008-01.

[23] N. Bruno and H. Wang, "The threshold algorithm: From middleware systems to the relational engine," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 4, pp. 523–537, 2007.

[24] X. Zhang and J. Chomicki, "On the semantics and evaluation of top-k queries in probabilistic databases," CSE Dept., University at Buffalo, SUNY, Tech. Rep. 2007-13.