Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2012) 1–21

Procedia Computer Science

Top-k best probability queries and semantics ranking properties on probabilistic databases

Trieu Minh Nhut Le, Jinli Cao, and Zhen He

trieule@sgu.edu.vn, j.cao@latrobe.edu.au, z.he@latrobe.edu.au

Abstract

There has been much interest in answering top-k queries on probabilistic data in various applications such as market analysis, personalised services, and decision making. In probabilistic relational databases, the most common problem in answering top-k queries (ranking queries) is selecting the top-k result based on scores and top-k probabilities. In this paper, we firstly propose novel answers to top-k best probability queries by selecting the probabilistic tuples which have not only the best top-k scores but also the best top-k probabilities. An efficient algorithm for top-k best probability queries is introduced without requiring users to define a threshold. The top-k best probability approach is a more efficient and effective than the probability threshold approach (PT-k) [1, 2]. Second, we add the *"k-best ranking score"* into the set of semantic properties for ranking queries on uncertain data proposed by [3, 4]. Then, our proposed method is analysed, which meets the semantic ranking properties on uncertain data. In addition, it proves that the answers to the top-k best probability queries overcome drawbacks of previous definitions of the top-k queries on probabilistic data in terms of semantic ranking properties. Lastly, we conduct an extensive experimental study verifying the effectiveness of answers to the top-k best probability queries compared to PT-k queries on uncertain data and the efficiency of our algorithm against the state-of-the-art execution of the PT-k algorithm using both real and synthetic data sets.

Keywords: Top-k Query, Ranking query, Probabilistic Data, Query Processing, Uncertain data.

1. Introduction

Uncertain data has arisen in some important applications such as personalized services, market analysis and decision making, because data sources of these applications are collected from data integration, data analysis, data statistics, data classification, and results prediction. These data are usually inconsistent [5] or contain likelihood information [6]. Thus, selecting the best choice from various alternatives of uncertain data is an important challenge facing these applications. The top-k queries that return the k best answers according to a user's function score are essential for exploring uncertain data on these applications [6]. Uncertain data have been studied extensively by many researchers in areas such as modelling uncertain data [7, 8], managing uncertain data [9], and mining uncertain data [10, 11].

1.1. Motivation

In business, investors often make decisions about their products based on analysis, statistical data and mining data [11], which provide predictions relating to successful and unsuccessful projects. To analyse the market, investors firstly collect the historical statistical data, and then use the data to predict future market trends with probabilistic prediction. This is known as probabilistic data. For example, assume that the data in Table 1 have been collected and analysed statistically, according to historical data resources [12]. Each tuple represents an investment project of USD $100 to produce a specific product (Product ID). Investing in products based on their probabilities (Probability) will result in an estimated amount of profit. In tuple t_1, a businessman invests USD $100 on product A, and it has a 0.29 chance of obtaining a profit of USD $25.

Tuple	Product ID	Profit of USD $100 investment	Probabilistic
t_1	A	25	0.29
t_2	B	18	0.3
t_3	E	17	0.8
t_4	B	13	0.4
t_5	C	12	1.0
t_6	E	11	0.2

Table 1. Predicted Profits of USD $100 investment on Products

In the real world, when analysing historical data, predictions on future market trends return two or more values per product with probabilities that the predictions are correct. Therefore, some tuples in Table 1 have the same product ID with different profit. In the probabilistic data model, these tuples are mutually exclusive, and controlled by a set of rules (generation rule) [1, 2, 6, 13]. For example, tuples t_2 and t_4 as project that invest in product B have a 0.3 probability of producing a USD $18 profit and 0.4 probability of producing a USD $13 profit. In this case, if the prediction for tuple t_2 is true, then the prediction for tuple t_4 will not be true. It is impossible for both profits to be true for the same product ID. They are mutually exclusive predictions. In Table 1, the probabilistic data are restricted by the exclusive rules $R_1 = t_2 \oplus t_4$ and $R_2 = t_3 \oplus t_6$.

Top-k queries can be used to help investors make business decisions such as choosing projects which have the top-2 highest profits. On probabilistic databases, top-k queries can be answered by using the probability space that enumerates the list of all possible worlds [1, 2, 7, 14, 15, 16]. A possible world contains a number of tuples in the probabilistic data set. Each possible world has a non-zero probability for existence and can contain k tuples with highest profits. Different possible worlds can contain different sets of k tuple answers. Therefore, it is necessary to list all possible worlds of Table 1 to find the top-2 answers for the top-2 query of the probabilistic database. Thus, Table 2 lists three dimensions: the possible world, the probability of existence, and the top-2 tuples in each possible world.

Possible world	Probability of existence	Top-2 tuples in possible world
$W_1 = \{t_1, t_2, t_3, t_5\}$	0.0696	t_1, t_2
$W_2 = \{t_1, t_2, t_5, t_6\}$	0.0174	t_1, t_2
$W_3 = \{t_1, t_3, t_4, t_5\}$	0.0928	t_1, t_3
$W_4 = \{t_1, t_4, t_5, t_6\}$	0.0232	t_1, t_4
$W_5 = \{t_1, t_3, t_5\}$	0.0696	t_1, t_3
$W_6 = \{t_1, t_5, t_6\}$	0.0174	t_1, t_5
$W_7 = \{t_2, t_3, t_5\}$	0.1704	t_2, t_3
$W_8 = \{t_2, t_5, t_6\}$	0.0426	t_2, t_5
$W_9 = \{t_3, t_4, t_5\}$	0.2272	t_3, t_4
$W_{10} = \{t_4, t_5, t_6\}$	0.0568	t_4, t_5
$W_{11} = \{t_3, t_5\}$	0.01704	t_3, t_5
$W_{12} = \{t_5, t_6\}$	0.0426	t_5, t_6

Table 2. List of all possible worlds and top-2 tuples

According to Table 2, any tuple has a probability of being in the top-2. Therefore, Table 3 lists the tuples, profit, probability, and top-2 probability to analyse the top-2 answers of probabilistic databases. The top-2 probability of a tuple is aggregated by the sum of its probabilities of existence in the top-2 in Table 2.

In previous research [1, 2], the top-k answers are found using the probability threshold approach called PT-k. The PT-k queries return a set of tuples with top-k probabilities greater than the users' threshold value. For example the

Tuple	Profit	Probability	Top-2 probability
t_1	25	0.29	0.29
t_2	18	0.3	0.3
t_3	17	0.8	0.7304
t_4	13	0.4	0.3072
t_5	12	1.0	0.3298
t_6	11	0.2	0.0426

Table 3. Top-2 probabilities

answer to the PT-2 query with threshold 0.3 in the example listed in Table 3 is the set containing 4 tuples $\{t_2, t_3, t_4, t_5\}$. We have identified three drawbacks with PT-*k* queries. These are listed below:

- The PT-*k* queries may lose some important results. According to PT-2 query, tuple t_1(25, 0.29) is eliminated by the PT-2 algorithm because its top-2 probability is less than threshold 0.3. In this case, we recommend that tuple t_1 should be in the result, the reason being that tuple t_1(25, 0.29) is not worse than tuple t_4(13, 0.3072), when comparing both attributes of profit and top-2 probability. That is, t_1.profit (25) is significantly greater than t_4.profit (13) and t_1.top-2 probability (0.29) is slightly less than t_2.top-2 probability (0.3072). In business, investors may like to choose project t_1 because they can earn nearly double the profit compared to project t_4, while project t_1 is only slightly riskier than project t_4 with a top-2 probability of 0.0172. Therefore, t_1 should be acceptable in the top-*k* answers.

- The PT-*k* answers may contain some redundant tuples which should be eliminated earlier in the top-*k* results. Referring to Table 3 again, tuples t_4 and t_5 should be eliminated immediately from the answer because the values of both attributes, profit and top-2 probability in t_4(13, 0.3072) and t_5(12, 0.3298), are less than those in t_3(18, 0.7304). It is obvious that investors will choose project t_3 which is more dominant than projects t_4 and t_5 in both profit and top-*k* probability.

- It is difficult for users to choose a threshold for the PT-*k* method. The threshold is a crucial factor used for efficiency and effectiveness in PT-*k* queries [1, 2]. Users may not know much about the probabilistic values and probabilistic databases. Therefore, it may be difficult for users to find the most suitable threshold initially. This, in turn, means they may need to waste time using trial and error to find the most suitable threshold value.

In addition, almost all previous top-*k* approaches such as U-top-*k* [14] and u-popk [16] focus on the highest probabilities relevant to some specific possible worlds as semantics of probabilistic databases [6] in order to select the top-*k* answers. With the aforementioned observations, there is a need to study the top-*k* query answers for a better solution. It is necessary to develop new answer to top-*k* query which will take both *top-k profit* and *top-k probability* on all possible worlds into account to select the best top-*k* answer. To address this concern, we propose new approach of the *"top-k best probability query"*. The results of top-*k* best probability queries have the following desirable properties:

1. Inclusion of important tuples: users are usually interested in the highest profits of projects. Therefore, the *k*-tuples with the highest profits (scores) should be in the answer set of the top-*k* queries on probabilistic data. For example, in Table 1, tuples t_1 and t_2 have the top-2 best profits and therefore should be in the top-2 answer.
2. Elimination of redundant tuples: the dominating concept on score and top-*k* probability of tuples is used to eliminate redundant tuples. The non-top-*k* score tuples are processed by the dominating concept to select the non-dominated tuples on score and top-*k* probability, which are added into the answer set of top-*k* queries. The non-dominated tuples will have *"the best top-k probabilities"*. For example, tuple t_3 is a non-dominated tuple which is added into the result because it has greater top-2 probability than a tuple in the top-2-highest scoring tuples. Then, the rest of the tuples t_4, t_5, t_6 in the data set are eliminated because they are dominated by tuple t_3 in both profit and top-2 probability.
3. Removal of the unclear threshold: The new effective method which combines the two previous techniques will remove the need for the threshold for processing the top-*k* queries on probabilistic databases.

Therefore, the set $\{t_1, t_2, t_3\}$ is an answer to the top-2 query on Table 1. The top-2 best probability query returns not only the tuples with the best top-2 ranking scores but also the top-2 highest probabilities to the users.

To create a new method of the top-*k* best probability query, which require more efficient and effective than previous work, is challenging because the answer must include the *k*-highest ranking scores tuples and eliminate redundant tuples without a threshold. In this paper, we will propose a new method for the top-*k* best probability query including definition, formulas, and algorithm to overcome these challenges. We also introduce a new property *"k-best ranking scores"* and add that into the semantic ranking properties on uncertain data. Those properties has been proposed to evaluate the semantics of ranking queries on uncertain data [3, 4]. Then, we prove whether the answers to top-*k* best probability queries and previous top-*k* queries satisfy these properties or not. It can be clearly seen that the results of the top-*k* best probability queries satisfy more semantic ranking properties than other existing top-*k* queries.

1.2. Contributions

Our contributions are summarized as follows:

- We introduce the new definition of the top-*k* best probability query on probabilistic databases, based on traditional top-*k* answers and a dominating concept, in which the dominating concept takes both the ranking score and top-*k* probability into account for selecting the top-*k* best probability tuples. We also develop formulas to calculate the top-*k* probability and handle the inclusive and exclusive rules (generation rule) on probabilistic databases.
- To improve the effectiveness of the algorithm of top-*k* best probability queries, some pruning rules are introduced. Their correctness is mathematically proven. These rules will be used to reduce the computation cost of top-*k* probabilities of tuples and allowing early stopping conditions to be defined. Then, the top-*k* best probability algorithm demonstrates that the answers to top-*k* best probability queries are more efficient than the answers to PT-*k* under various probabilistic data sources.
- We prove that our top-*k* best probability queries cover the existing semantic properties from [3, 4] and also covers a new property proposed in this paper called *"k-best ranking scores"*. This property is introduced according to the users' expectations.
- Both real and synthetic data sets are used in our extensive experimental study to evaluate the proposed approach. The PT-*k* method is compared with the top-*k* best probability method in terms of effectiveness and efficiency.

This paper is an extended version of our earlier paper [17], in which we proposed the top-*k* best probability query to improve efficiency of algorithm and effectiveness of the answers. In this paper, we extend the top-*k* best probability query on probabilistic data model containing inclusive rules and add a new pruning rule to improve the efficiency of the top-*k* best probability algorithm. As a result, answers to the top-*k* best probability queries also cover more semantic ranking properties than other answers to top-*k* queries. We have performed additional experiments which show the effectiveness and efficiency of our proposed method.

The rest of the paper is organized as follows. Section 2 presents the preliminaries in which we review the probabilistic database model, definition, its components, and computations of top-*k* probability of tuples to support our approach. In Section 3, we introduce the dominating concept on score and top-*k* probability of tuple, and propose the novel approach of the *top-k best probability query* on probabilistic data including a formal definition, pruning rules, and an efficient algorithm. The next section formulates the previous properties of semantic ranking queries, and proposes the new property *k-best ranking scores*, which are then used to evaluate whether the semantics of ranking queries on probabilistic data have been satisfied or not. In Section 5, extensive experiments using real and synthetic data are conducted to show the effectiveness and efficiency of the top-*k* best probability method. Finally, section 6 briefly concludes our contributions and outlines future research.

2. Preliminary of probabilistic databases and top-k queries

In this section, several definitions and models for probabilistic data processing on probabilistic databases are presented formally, which are similar to papers [18, 1, 2, 9, 19, 10, 20, 21, 22, 16, 23]. We review notions and the foundation of the probabilistic database model. In this model, the process of calculating the top-*k* probability tuples is also addressed for our proposal.

2.1. Probabilistic database model

Generally, the probabilistic database $\mathcal{D} = \{t_1, ..., t_n\}$ is assumed to be a single probabilistic table, which is a finite set of probabilistic tuples, where each probabilistic tuple is a tuple associated with probability to denote the uncertainty of the data, and almost all probabilistic tuples in probabilistic databases are independent.

Definition 1. *Probability of a tuple is the likelihood of a tuple appearing in the data set $\mathcal{D}$. The probability $p(t_i)$ of tuple t_i can be presented as a new dimension in the table with values of* $0 < p(t_i) \leq 1$.

Example: In Table 1, tuple t_1 has a 0.29 probability of obtaining a USD$ 25 profit.

Based on the probability theory, if the probability of tuple t_i is $p(t_i)$, the probability $1 - p(t_i)$ is the unlikelihood of t_i in the data.

A simple way to study probabilistic data is listing sets of tuples, in which each tuple is present or absent corresponding to specific probabilities which are called possible worlds. It is interesting to view all possible worlds and study various solutions on probabilistic data.

Definition 2. *A possible world represents the semantics of a probabilistic database. Each possible world $W_j = \{t_1, ..., t_k\}$ contains a number of tuples which are members of probabilistic database $\mathcal{D}$. Each possible world is associated with a probability to indicate its existence. The probability of the existence of possible world W_j is calculated by multiplying the probabilities of the tuples, which are the likelihood and unlikelihood of tuples in the possible world W_j. All possible worlds $\mathcal{W} = \{W_1, ..., W_m\}$ is called the possible world space.*

Example: Table 2 is the possible world space that lists all possible worlds of Table 1.

For realistic situations, probabilistic tuples on probabilistic data can be independent or dependent on each others. Several probabilistic tuples can be present together or restrict each other's presence in the possible worlds. On the probabilistic database model, they are grouped as a generation rule.

Definition 3. *In probabilistic data, a generation rule is a set of exclusive or inclusive rules. Each rule contains special tuples, which can be mutually exclusive or inclusive in the possible world space.*

- *The exclusive rule is in the form of $R^+_h = t_{h_1} \oplus t_{h_2} \oplus ... \oplus t_{h_q}$, where $\oplus$ is an exclusive operator, and $t_{h_1}, t_{h_2}, ..., t_{h_q}$ are members of probabilistic data, which indicates that, at most, one tuple can appear in the same possible world. The sum of all probabilities of tuples in the same exclusive rule must be less than or equal to 1, $p(R^+_h) = \sum_{i=1}^{q} p(t_{h_i}) \leq 1$.*
- *The inclusive rule is in the form of $R^*_{h'} = t_{h'_1} \wedge t_{h'_2} \wedge ... \wedge t_{h'_{q'}}$. where $\wedge$ is an inclusive operator, and $t_{h'_1}, t_{h'_2}, ..., t_{h'_{q'}}$ are members of probabilistic data, which indicates that non-tuple or all tuples appear in the same possible world. Tuples in $R^*_{h'}$ have the same probability $p(R^*_{h'}) = p(t_{h'_1}) = p(t_{h'_2}) = ... = p(t_{h'_{q'}})$ $\{0 < p(R^*_{h'}) \leq 1\}$.*

Example: In Table 1, the generation rule contains two exclusive rules and zero inclusive rule. Exclusive rules $R^+_1 = t_2 \oplus t_4$ and $R^+_2 = t_3 \oplus t_6$. R^+_1 indicates that tuples t_2 and t_4 never appear together in the same possible world. This is the same for tuples t_3 and t_6 in R^+_2.

2.2. Calculation of top-k probability

In this subsection, we discuss and formulate the calculation of top-k probabilities of tuples in the possible world space.

Definition 4. *The top-k probability of a tuple $pr^k_{top}(t_i)$ is the likelihood of tuple t_i appearing in the top-k in all possible worlds. Mathematically, the top-k probability of a tuple $pr^k_{top}(t_i)$ is the sum of the probabilities of the existence of possible worlds in which the tuple is in the top-k.*

Example: In Table 2, since t_2 is one of the top-2 tuples in possible worlds (W_1, W_2, W_7, W_8), the top-2 probability of tuple t_2 is the sum of the probabilities of the existence of these possible worlds. That is, $pr^2_{top}(t_2) = 0.0696 + 0.0174 + 0.1704 + 0.0426 = 0.3$.

In the probabilistic data, when the number of tuples is increasing, it is impossible to list all the possible worlds and calculate every probability of the existence of possible worlds at a limited time, because the number of all the possible worlds is 2^n and the computation cost of all probabilities of its existence is very expensive. It is impractical to calculate the top-k probability based on listing the possible world space. Therefore, formulas to calculate the top-k probability of tuples are required. The formulas for this problem have already been solved in mathematics, so we will adapt these formulas to this research.

Let probabilistic data $\mathcal{D}$ be a ranked sequence $(t_1, t_2, ..., t_n)$ with $t_1.score \preceq t_2.score \preceq ... \preceq t_n.score$, and $S_{t_i} = (t_1, t_2, ..., t_i)$ $(1 \leq i \leq n)$ be a subsequence from t_1 to t_i.

In the possible world space, the top-k probability of tuple t_i is the sum of all probabilities of t_i being ranked from 1 to k. When tuple t_i is ranked at the $(j+1)^{th}$ position, there are also j tuples in the subsequence set $S_{t_i} = (t_1, t_2, ..., t_i)$ appearing in possible worlds.

Theorem 1. *Given a ranked sequence $S_{t_i} = (t_1, t_2, ..., t_i)$, $pr(S_{t_i}, j)$ is the probability of any j tuples $(0 < j \leq i < n)$ appearing in the sequence S_{t_i}. $pr(S_{t_i}, j)$ is calculated as follows:*

$$pr(S_{t_i}, j) = pr(S_{t_{i-1}}, j-1) \times p(t_i) + pr(S_{t_{i-1}}, j) \times (1 - p(t_i))$$

In special cases

- $pr(\phi, 0) = 1$
- $pr(\phi, j) = 0$
- $pr(S_{t_{i-1}}, 0) = \prod_{j=1}^{i} (1 - p(t_j))$

This Poisson binomial recurrence has been proven in [24].

Example: Given the data set in Table 1, $pr(S_{t_1}, 1)$, the probability of any one tuple appearing in the sequence $S_{t_1} = \{t_1\}$ and $pr(S_{t_2}, 1)$, the probability of any one tuple appearing in the sequence $S_{t_2} = \{t_1, t_2\}$ are computed as follows:

$$\begin{aligned} pr(S_{t_1}, 1) &= pr(\phi, 0) \times p(t_1) + pr(\phi, 1) \times (1 - p(t_1)) \\ &= 1 \times 0.29 + 0 \times (1 - 0.29) = 0.29 \\ pr(S_{t_2}, 1) &= pr(S_{t_1}, 0) \times p(t_2) + pr(S_{t_1}, 1) \times (1 - p(t_2)) \\ &= (1 - 0.29) \times 0.3 + 0.29 \times (1 - 0.3) = 0.416 \end{aligned}$$

Based on Theorem 1, the probability of a tuple being ranked at the exact position is represented by the follow theorem.

Theorem 2. *Suppose that tuple t_i and the subsequence $S_{t_{i-1}}$, $pr(t_i, j)$ is the probability of tuple t_i which is ranked at the exact j^{th} position $(n > i \geq j > 0)$. $pr(t_i, j)$ is calculated as follows:*

$$pr(t_i, j) = p(t_i) \times pr(S_{t_{i-1}}, j-1)$$

This Poisson binomial recurrence has been proven in [24].

Example: Given the data set in Table 1, the 1^{st} and 2^{nd} rank probabilities of tuple t_3 are computed as follows:

$$\begin{aligned} pr(t_3, 1) &= p(t_3) \times pr(S_{t_2}, 0) \\ &= 0.8 \times (1 - 0.29) \times (1 - 0.3) = 0.3976 \\ pr(t_3, 2) &= p(t_3) \times pr(S_{t_2}, 1) \\ &= 0.8 \times 0.416 = 0.3328 \end{aligned}$$

To calculate the top-k probability of tuple t_i, Theorem 3 is extended from Theorem 2, that is, a sum of ranked positions from 1^{st} to k^{th} probabilities of tuple t_i.

Theorem 3. *Given a tuple t_i, $pr^k_{top}(t_i)$ is the top-k probability of t_i in the possible world space, then $pr^k_{top}(t_i)$ is calculated as follows:*

$$pr^k_{top}(t_i) = \sum_{j=1}^{k} pr(t_i, j) = p(t_i) \times \sum_{j=1}^{k} pr(S_{t_{i-1}}, j-1)$$
$$If\ i \leq k\ then\ pr^k_{top}(t_i) = p(t_i)$$

Example: Given the data set in Table 1, the top-2 probability of tuple t_3 is computed as follows:

$$\begin{aligned} pr^2_{top}(t_3) &= \sum_{j=1}^{2} pr(t_3, j) \\ &= pr(t_3, 1) + pr(t_3, 2) \\ &= 0.3976 + 0.3328 = 0.7304 \end{aligned}$$

In general, the above three theorems are used to directly calculate the top-k probability of all tuples without listing any possible worlds.

2.3. Calculation of top-k probability with a generation rule

Generally, we can use the above theorems to calculate the top-k probability of each tuple. However, the probabilistic data model involves mutually exclusive and inclusive rules which have to be mentioned when calculating the top-k probability of tuple formulas. Therefore, the calculations of top-k probabilities have to take these rules into account.

Let a probabilistic database $\mathcal{D} = (t_1, t_2, ..., t_n)$ with $t_1.score \leq t_2.score \leq ... \leq t_n.score$ be ranked as the sequence, we follow paper [1, 2] for processing the exclusive rules and inclusive rules in formulas to calculate the top-k probabilities of tuple t_i.

2.3.1. Exclusive rules

Tuples in the same rule are mutually exclusive $R^+_h = t_{h_1} \oplus ... \oplus t_{h_m}$, which indicates that, at most, one tuple can appear in the same possible worlds. Therefore, an exclusive rule can be produced as a tuple. Then, the previous formulas are modified to calculate the top-k probability.

To compute the top-k probability $pr^k_{top}(t_i)$ of tuple $t_i \in \mathcal{D}\,(1 \leq i \leq n)$, t_i divides the generation rule $R^+_h = t_{h_1} \oplus ... \oplus t_{h_m}$ into two parts $R^+_{hLeft} = t_{h_1} \oplus ... \oplus t_{h_j}$ and $R^+_{hRight} = t_{h_{j+1}} \oplus ... \oplus t_{h_m}$. The tuples involved in R^+_{hLeft} are ranked higher than or equal to tuple t_i. The tuples involved in R^+_{hRight} are ranked lower than tuple t_i. According to this division, the following cases demonstrate the exclusive rule produced as a tuple into the formulas to calculate the top-k probability of tuple t_i.
- Case 1: $R^+_{hLeft} = \phi$, i.e. all the tuples in rule R^+_h are ranked lower than tuple t_i. Therefore, all tuples in R^+_h are not considered when calculating the top-k probability of tuple t_i. Consequently, all tuples in R^+_h are ignored.
- Case 2: $R^+_{hLeft} \neq \phi$, i.e. all tuples in R^+_{hRight} can be ignored, and the tuples in R^+_{hLeft} have been changed when calculating the top-k probability of t_i. There are two sub-cases for these changes.

+ sub-case 1: $t_i \in R^+_{hLeft}$, i.e. t_i has already appeared in the possible world space. The other tuples in R^+_{hLeft} can not appear, so these tuples will be removed from subsequence $S_{t_{i-1}}$ when calculating $pr^k_{top}(t_i)$.

+ sub-case 2: $t_i \notin R^+_{hLeft}$, i.e. all tuples in R^+_{hLeft} will be produced and considered as a tuple $t_{R^+_{hLeft}}$ with their sum probabilities $p(t_{R^+_{hLeft}}) = \sum_{R^+_{hLeft}} p(t_{hLeft})$.

After all the exclusive rules are produced, the formulas for calculating the top-k probability in the previously mentioned theorems are used normally.

Example: In Table 1, the top-2 probability of tuples t_6 $pr^2_{top}(t_6)$ will be calculated by applying two exclusive rules $R^+_1 = t_2 \oplus t_4$ and $R^+_2 = t_3 \oplus t_6$. The subsequence S_{t_6} contains the tuples $\{t_1, t_2, t_3, t_4, t_5, t_6\}$ with their probabilities $\{0.29, 0.3, 0.8, 0.4, 1.0, 0.2\}$, respectively.

The exclusive rules are produced in subsequence S_{t_6}.

- In exclusive rule R^+_2, t_3 is removed because t_3 and t_6 are in the same exclusive rule (sub-case 1 of exclusive rule)

- In exclusive rule R_1^+, $t_2 \oplus t_4$ are produced as $t_{2\oplus4}$. The probability of $t_{2\oplus4}$ is 0.7 (sub-case 2 of exclusive rule) The subsequence S_{t_6} is produced by the exclusive rule $\{t_1, t_{2\oplus4}, t_5, t_6\}$ with their probabilities $\{0.29, 0.7, 1.0, 0.2\}$.

$pr_{top}^2(t_6)$ with the set $S_{t_5}\{t_1, t_{2\oplus4}, t_5\}$ is calculated as follows:

$$pr_{top}^2(t_6) = p(t_6) \times \sum_{j=1}^{2}(S_{t_5}, j-1)$$

$$pr_{top}^2(t_6) = p(t_6) \times (pr(S_{t_5}, 0) + pr(S_{t_5}, 1)), \text{ where}$$

$$pr(S_{t_5}, 0) = (1-0.29) \times (1-0.7) \times (1-1) = 0$$

$$pr(S_{t_5}, 1) = (1-0.29) \times (1-0.7) = 0.213$$

$$pr_{top}^2(t_6) = 0.2 \times (0 + 0.213) = 0.0426$$

2.3.2. Inclusive rules

Tuples in the same rule are mutually inclusive $R_{h'}^* = t_{h'_1} \wedge t_{h'_2} \wedge ... \wedge t_{h'_{q'}}$, which indicates that no tuple or all tuples appear in possible worlds. To compute the top-k probability $pr_{top}^k(t_i)$ of tuple $t_i \in \mathcal{D}(1 \le i \le n)$, tuple t_i can be in an inclusive rule or not in the following cases.

- Case 1: All the tuples in rule $R_{h'}^*$ are ranked lower than tuple t_i. All tuples in $R_{h'}^*$ are ignored.
- Case 2: At least one tuple in inclusive rule $R_{h'}^*$ is ranked higher than or equal to tuple t_i. There are two sub-cases.

+ sub-case 1: $t_i \in R_{h'}^*$, i.e. t_i has already appeared in the possible world space, and there are z tuples ranked higher than t_i in $R_{h'}^*$ $(z < k)$, which also appeared with t_i in the same possible worlds. Therefore, the top-k probability $pr_{top}^k(t_i)$ of tuple t_i of Theorem 3 is modified as follows:

$$pr_{top}^k(t_i) = p(t_i) \times \sum_{j=1}^{k-z} pr(S_{t_{i-1}} \setminus R_h^*, j-1)$$

+ sub-case 2: $t_i \notin R_{h'}^*$, i.e. there are z tuples ranked higher than t_i in $R_{h'}^*$ $(z < k)$, and these tuples are considered as a tuple $t_{R_{h'}^*}$ with probability $p(t_{R_{h'}^*}) = p(t_{h'_1}) = p(t_{h'_2}) = ... = p(t_{h'_{q'}})$. The sequence S_{t_i} now is $S'_{t_i} \cup \{t_{R_{h'}^*}\}$, where $S'_{t_i} = \left\{\{t_1, t_2, ..., t_i\} \setminus \{t' \mid t' \in R_{h'}^*\}\right\}$, then a probability of any j tuples $(0 < j \le i < n)$ appearing in the set S_{t_i} is modified as follows:

- ◦ $pr(S_{t_i}, 0) = pr(S'_{t_i}, 0) \times (1 - p(t_{R_{h'}^*}))$
- ◦ $pr(S_{t_i}, j) = pr(S'_{t_i}, j) \times (1 - p(t_{R_{h'}^*}))$ (for $0 < j < z$)
- ◦ $pr(S_{t_i}, j) = pr(S'_{t_i}, j-z) \times p(t_{R_{h'}^*}) + pr(S'_{t_i}, j) \times (1 - p(t_{R_{h'}^*}))$ (for $j \ge z$)

Example: Assuming two inclusive rules $R_1^* = t_3 \wedge t_5$ and $R_2^* = t_4 \wedge t_6$ with probabilities $p(R_1^*) = p(t_3) = p(t_5) = 0.7$ and $p(R_2^*) = p(t_4) = p(t_6) = 0.4$ are in probabilistic database $\mathcal{D} = (t_1, t_2, ..., t_{10})$ in ranking order. To compute $pr_{top}^4(t_6)$, the inclusive rules are produced in subsequence S_{t_5}.

- In inclusive rule $R_2^* = t_4 \wedge t_6$, $t_6 \in R_2^*$, sub-case 1 of the inclusive rule is applied. The subsequence $S_{t_5} \setminus R_2^*$ is $\{t_1, t_2, t_3, t_5\}$, and z=1 because t_4 has a higher rank than t_6. We compute the top-4 probability of tuple t_6

$$pr_{top}^4(t_6) = p(t_6) \times \sum_{j=1}^{4-1} pr(\{t_1, t_2, t_3, t_5\}, j-1)$$

- In inclusive rule $R_1^* = t_3 \wedge t_5$, tuples t_3, t_5 are ranked higher than tuple t_6. We produced inclusive rule R_1^* into $t_{3,5}$ with probability $p(t_{3,5}) = p(R_1^*) = 0.7$. The set S_{t_5} is $\{t_1, t_2, t_{3,5}\}$, and the set $S'_{t_5} = \{S_{t_5} \setminus t_{3,5}\}$ is $\{t_1, t_2\}$ or $S'_{t_5} = S_{t_2}$.

Then, sub-case 2 of the inclusive rule is applied to calculate the top-4 probability of tuple t_6.

$$pr^4_{top}(t_6) = p(t_6) \times \sum_{j=1}^{3} pr(\{t_1, t_2, t_{3,5}\}, j-1)$$

$$pr^4_{top}(t_6) = p(t_6) \times (pr(\{t_1, t_2, t_{3,5}\}, 0) + pr(\{t_1, t_2, t_{3,5}\}, 1) + pr(\{t_1, t_2, t_{3,5}\}, 2))$$

$$pr(\{t_1, t_2, t_{3,5}\}, 0) = pr(S_{t_2}, 0) \times (1 - p(t_{3,5}))$$

$$pr(\{t_1, t_2, t_{3,5}\}, 1) = pr(S_{t_2}, 1) \times (1 - p(t_{3,5}))$$

$$pr(\{t_1, t_2, t_{3,5}\}, 2) = pr(S_{t_2}, 0) \times p(t_{3,5}) + pr(S_{t_2}, 2) \times (1 - p(t_{3,5}))$$

3. The top-k best probability queries

This section presents the proposed method for finding and computing the answer to the top-*k* best probability query. First, we create a new definition of the top-*k* best probability query which is based on the rationale described in the introduction. Secondly, we present the significance of our proposed method. Then, we introduce the process to select an answer to the top-*k* best probabilistic queries, and several pruning rules for an effective algorithm. Lastly, we describe the algorithm for computing the top-*k* best probability query.

3.1. Definition of the top-k best probability

Tuples which are the result of probabilistic top-*k* queries must consider not only the ranking score but also the top-*k* probability [25, 21]. In the area of probabilistic data, significant research is being conducted on the semantics of top-*k* queries. However, the semantics between high scoring tuples and high top-*k* probabilities of tuples is interpreted differently by various researchers. Our ranking approach considers both dimensions of ranking score and top-*k* probability independently, in which the ranking score cannot be considered more important than the top-*k* probability and vice versa.

To answer a probabilistic query, every tuple has two associated dimensions: the top-*k* probability and its ranking score. These two dimensions are crucial for choosing the answer to the top-*k* query on probabilistic data. They are also independent of real world applications. In this research, we introduce the concept of dominating tuples to select the full meaning of top-*k* tuples which are non-dominated tuples. This concept is widely used for multiple independent dimensions for skyline queries in many papers [26, 22, 27, 28, 29, 30]. In this paper, the domination concept for top-*k* queries is used to compare top-*k* tuples in the two dimensions of score and top-*k* probability.

Definition 5. *(domination of top-k probability tuples) For any tuples t_i and t_j in probabilistic data, t_i dominates t_j ($t_i \prec t_j$), if and only if t_i has been ranked higher than t_j and the top-k probability of t_i is greater than the top-k probability of t_j ($pr^k_{top}(t_i) > pr^k_{top}(t_j)$), else t_i does not dominate t_j ($t_i \nprec t_j$).*

Example: in Table 3, tuple t_3 dominates t_5, because t_3 has a higher rank than t_5 ($rank(t_3.score) = 3, rank(t_5.score) = 5$) and top-2 probability (0.7304) of t_3 is greater than top-2 probability (0.3072) of t_5.

Definition 6. *(non-dominated tuple) a tuple which is not dominated by all the other tuples on score and top-k probability is the non-dominated tuple.*

Example: in Table 3, tuples t_1 and t_3 are the non-dominated tuples in the set $\{t_1, t_3, t_4, t_5, t_6\}$.

In previous research, the answer to PT-*k* is based on the top-*k* probabilities and the threshold. The resultant set, which is analyzed in the introduction, may exclude some highest ranking score tuples or include some redundant tuples. Therefore, we have to create new top-*k* queries which overcome these problems. Definition 7 is introduced to select the best tuples on score and top-*k* probability using the domination concept to improve the quality of the top-*k* answers. That is, we are looking at tuples which are in both the top-*k* best ranking scores and the best top-*k* probabilities.

Definition 7. *(top-k best probability query) The answer to the top-k best probability query Q^k_{top} consists of two sets Q_{score} and Q_{pro} where Q_{score} contains the top-k ranking score tuples in the data without considering the top-k probabilities, and Q_{pro} contains non-dominated tuples on score and top-k probability in the set $\{\{\mathcal{D} \setminus Q_{score}\} \cup \{t_{pmin}\}\}$, where t_{pmin} is the tuple with the lowest top-k probability in Q_{score}.*

The top-*k* best probability query can overcome the problems by selecting all the tuples with the highest ranking score for set Q_{score}, and remove the redundant tuples based on the non-dominated tuples concept for set Q_{pro}. The set Q_{score} contains the top-*k* highest scores tuples, therefore t_{pmin} is selected as the lowest top-*k* probability in Q_{score} to find the non-dominated tuples which have better top-*k* probabilities in the rest of tuples $\{\mathcal{D} \setminus Q_{score}\}$.

For example, in Table 1, the process to obtain the final answer to top-2 best probability query is described in Definition 7. The answer set to the top-2 best probability query is $Q^2_{top} = \{t_1, t_2, t_3\}$, in which Q_{score} $\{t_1, t_2\}$ is the top-2 highest ranked profit tuples without considering the top-2 probabilities (the common result), and Q_{pro} $\{t_1, t_3\}$ contains the non-dominated tuples on score and top-*k* probability in $\{t_1, t_3, t_4, t_5, t_6\}$, where t_1 is t_{pmin}.

3.2. Significance of top-k best probability query

We now discuss the top-*k* best probability query, matching the proposed requirements, then compare our results to the previous proposals.

3.2.1. k-highest ranking score tuples and dominating concept for semantic answers

Generally, traditional top-*k* queries on databases return the *k*-highest ranking score tuples. However, almost all previous papers on top-*k* queries on probabilistic data selected top-*k* tuples based on some specific possible worlds with scores such as U-top-*k* [14], U-*k*-ranks [31], and U-popk [16], or tuples with high values combining scores and probabilities such as E-score [3] and E-rank [3]. However, these previous approaches lose some important top-*k* ranking score tuples. Therefore, the top-*k* best probability query must firstly select the *k*-best ranking score tuples of the database, ignoring its probability. This property allows the top-*k* best probability query to overcome the information loss problem which is inherent in traditional top-*k* queries.

The dominating concept is used to find the non-dominated tuples for the best top-*k* probability value. This is the key to our proposal which has solved almost all the limitations of the previous approaches. The dominating concept also takes both top-*k* ranking score and top-*k* probability into account. A number of papers have discussed probabilistic data queries. The semantics of answering a query can have different meanings in various applications [10, 32, 33]. The U-top-*k* [14], C-Typical-top-*k* [18], and E-rank [3] tried to create relations between the ranking scores and top-*k* probabilities on probabilistic data in different domains. These approaches are suitable in their specific situations, because these relations have different meanings semantically. However, the real semantics of top-*k* answers are mainly based on the users' decisions on both top-*k* ranking scores and top-*k* probabilities. It is appropriate to use the *non-dominated concept* for selecting answers to users.

3.2.2. Threshold vs. bestPr

A threshold plays an important role in the PT-*k* algorithm in paper [1, 2]. The authors used the user define static threshold to prune irrelevant answers. Hence, the PT-*k* algorithm of top-*k* queries seem to be more effective and efficient for processing the top-*k* answers. However, it could be ineffective and inefficient for users, because the problem is the selecting a threshold which is set by users. The users randomly select the threshold from 0 to 1 several times until obtaining a satisfactory answer. If the users choose a lower threshold than the top-*k* probabilities of tuples, the computation cost of the PT-*k* algorithm to the top-*k* queries will be expensive due to the need to calculate almost all the top-*k* probabilities of tuples. Otherwise, if the users choose a higher threshold than top-*k* probabilities, the result set of the PT-*k* algorithm will be empty due to pruning all top-*k* probability tuples by the higher threshold. Moreover, users sometimes do not have enough information to set the threshold for the probabilistic data. Therefore, it is not easy to choose a suitable threshold value as the threshold is an unclear value for the users. Therefore, based on the observation, our proposed algorithm does not require users to provide a threshold. Our algorithm uses *bestPr* which is automatically initialized and updated during execution for effective pruning.

3.3. Finding top-k best probability and pruning rules

We now describe a technique for selecting the top-*k* best probability answers, and present effective pruning rules for the top-*k* best probability algorithm.

a. Selecting the answer to the top-k best probability queries

Suppose that the probabilistic data set has been ranked by score, we have divided n probabilistic tuples of probabilistic data set $\mathcal{D}$ into two sets $Q_{score} = \{t_1, t_2, ..., t_k\}$ and $\mathcal{D} \setminus Q_{score} = \{t_{k+1}, t_{k+2}, ..., t_n\}$. Q_{score} contains the first k-highest ranking score tuples.

To select all non-dominated tuples for Q_{pro}, we first pick the tuple t_{pmin} which has the lowest top-k probability in Q_{score}. This lowest top-k probability tuple is always the first non-dominated tuple in $\{Q_{pro} \cup \{t_{pmin}\}\}$, because t_{pmin} has a higher rank than all other tuples in $\mathcal{D} \setminus Q_{score}$. This means that all tuples in $\mathcal{D} \setminus Q_{score}$ do not dominate t_{pmin}. As we already have the tuples ranking order, the rest of the non-dominated tuples are selected by only considering their top-k probabilities. The initial value $bestPr$ is assigned $bestPr = \min_{i=1}^{k}(pr^k_{top}(t_i))$. To select the non-dominated tuples from $\{t_{k+1}, t_{k+2}, ..., t_n\}$, the top-$k$ probability of each tuple from t_{k+1} to t_n is calculated in succession. In each tuple, the top-k probability will be compared to $bestPr$. If it is greater than $bestPr$, the tuple will be inserted into Q_{pro}, and $bestPr$ is assigned a new value which is the greater top-k probability. The inserted tuple is non-dominated in Q_{pro} because its top-k probability is greater than all top-k probabilities of tuples in Q_{pro}. When all tuples are executed, all the non-dominated tuples of Q_{pro} are found. The answer set $Q^k_{top} = Q_{score} \cup Q_{pro}$ is returned for the top-k best probability query.

The value of $bestPr$ will be increased while selecting the non-dominated tuples, which have higher top-k probabilities in the answer L_{pro}, therefore $bestPr$ is called the *best top-k probability*. The $bestPr$ is also the key value to improve the effectiveness and efficiency of our proposed algorithm because it is used to eliminate all tuples with low top-k probabilities without the need to conduct calculations by following the pruning rules.

b. Pruning rules

In this subsection, we introduce and prove several theorems to minimize the calculation of the top-k probability of tuples and activate the stopping condition which negates the need to calculate the remaining tuples.

The first pruning rule is presented as Theorem 4 which uses the $bestPr$ value to eliminate the current tuple without calculating its top-k probability.

Theorem 4. *Given any tuple t_i and bestPr, if $p(t_i) \le bestPr$ then $pr^k_{top}(t_i) \le bestPr$*

Proof. The top-k probability of t_i is calculated by Theorem 3

$$pr^k_{top}(t_i) = p(t_i) \times \sum_{j=1}^{k} pr(S_{t_{i-1}}, j-1)$$

with $p(t_i) \le bestPr$, and $\sum_{j=1}^{k} pr(S_{t_{i-1}}, j-1) \le 1$ then

$$pr^k_{top}(t_i) = p(t_i) \times \sum_{j=1}^{k} pr(S_{t_{i-1}}, j-1) \le bestPr$$

□

Theorem 5 is a pruning rule which uses probability and top-k probabilities of previous tuples to eliminate the current tuple without the need to perform top-k probability calculation

Theorem 5. *Given any t_{i-m}, t_i and bestPr, if $p(t_{i-m}) \ge p(t_i)$ and $pr^k_{top}(t_{i-m}) \le bestPr$ then $pr^k_{top}(t_i) < bestPr$ for any $1 \le m < i$*

Proof. The top-k probability of t_{i-m} is calculated by Theorem 3

$$pr^k_{top}(t_{i-m}) = p(t_{i-m}) \times \sum_{j=1}^{k} pr(S_{t_{i-m-1}}, j-1)$$

For any $1 \leq m < i$, we have sequence $S_{t_{i-m}} \subset S_{t_i}$, it is also true when S contains generation rules.

$$\sum_{j=1}^{k} pr(S_{t_{i-m-1}}, j-1) \geq \sum_{j=1}^{k} pr(S_{t_{i-1}}, j-1)$$

with $pr^k_{top}(t_{i-m}) \leq bestPr$, and $p(t_{i-m}) > p(t_i)$ then

$$pr^k_{top}(t_i) = p(t_i) \times \sum_{j=1}^{k} pr(S_{t_{i-1}}, j-1) \leq pr^k_{top}(t_{i-m}) \leq bestPr$$

□

We note that Theorem 5 is used in our algorithm, in which only the $p(t_{i-m}) \geq p(t_i)$ condition must be considered for eliminating the current tuple, because the other $pr^k_{top}(t_{i-m}) \leq bestPr$ condition is always true. The reason is because *bestPr* is assigned the best top-*k* probability value among tuples t_{k+1} to t_{i-1} during algorithm processing.

The following Theorem 6 is the most important rule in our algorithm. It is used to stop the algorithm processing and return the answer. The algorithm can stop early, before it calculates the top-*k* probabilities of the other lower ranking score tuples.

Theorem 6. *For any t_i and bestPr, if $bestPr \geq \sum_{j=1}^{k} pr(S_{t_i} \backslash \{t_{L_{max}}\}, j-1)$ then $bestPr \geq pr^k_{top}(t_{i+m})$ for any $m \geq 1$ where*
- *$S_{t_i} = (t_1, t_2, ..., t_i)$ is the ranked sequence.*
- *$t_{L_{max}}$ is the produced tuple of the generation rule which has the highest produced probability than the other produced probabilities in the generation rules.*

Proof. The top-*k* probability of t_{i+m} is calculated by Theorem 3

$$pr^k_{top}(t_{i+m}) = p(t_{i+m}) \times \sum_{j=1}^{k} pr(S_{t_{i+m-1}}, j-1)$$

In the worst case, t_{i+m} is in exclusive rule R_h.

$$pr^k_{top}(t_{i+m}) \leq p(t_{i+m}) \times \sum_{j=1}^{k} pr(S_{t_{i+m-1}} \backslash t_{R_{hLeft}}, j-1)$$

For any $m > 1$, we have sequence $S_{t_i} \subseteq S_{t_{i+m-1}}$, It is also true when calculating the $pr^k_{top}(t_i)$ with S'_{t_i} containing an inclusive rule.

$$pr^k_{top}(t_{i+m}) \leq p(t_{i+m}) \times \sum_{j=1}^{k} pr(S_{t_i} \backslash t_{R_{hLeft}}, j-1)$$

In exclusive rules, we have $t_{R_{hLeft}} \leq t_{L_{max}}$ for any produced tuples

$$pr^k_{top}(t_{i+m}) \leq p(t_{i+m}) \times \sum_{j=1}^{k} pr(S_{t_i} \backslash t_{L_{max}}, j-1)$$

For the probability of any tuple, we have $0 < p(t_{i+m}) \leq 1$

$$pr^k_{top}(t_{i+m}) \leq \sum_{j=1}^{k} pr(S_{t_i} \backslash t_{L_{max}}, j-1)$$

Hence, if $bestPr \geq \sum_{j=1}^{k} pr(S_{t_i} \backslash t_{L_{max}}, j-1)$, then $bestPr \geq pr^k_{top}(t_{i+m})$ □

```
Input: probabilistic data $\mathcal{D}$ in ranking order, generation rules $\mathcal{R}$.
Output: $Q^k_{top}$ answer set to top-$k$ best probability query
foreach (tuple $t_i$ {$i$=1 to $n$}) do
    if ($i \leq k$) then
        $pr^k_{top}(t_i) \leftarrow p(t_i)$ {special case in theorem 3};
        $Q^k_{top} \leftarrow Q^k_{top} \cup (t_i, pr^k_{top}(t_i))$;
        $bestPr \leftarrow \min(pr^k_{top}(t_i))$;
        $p_{prev} \leftarrow bestPr$;
    else
        if $(p(t_i) > bestPr)$ {Theorem 4} $\bigcap$ $(p(t_i) > p_{prev})$ {Theorem 5} then
            $pr^k_{top}(t_i) \leftarrow$ calculating top-k probability with generation rules $\mathcal{R}$ in Section 2 ;
            if $(pr^k_{top}(t_i) > bestPr)$ then
                $bestPr \leftarrow pr^k_{top}(t_i)$;
                $Q^k_{top} \leftarrow Q^k_{top} \cup (t_i, pr^k_{top}(t_i))$;
                $p_{prev} \leftarrow p(t_i)$;
        if (bestPr satisfies Theorem 6) then
            Exit;
```

Algorithm 1: The top-k best probability algorithm

3.4. The top-k best probability algorithm

We now propose the algorithm for selecting the answers to the top-*k* best probability queries on probabilistic data. Algorithm 1 finds the top-*k* best probability answer $Q^k_{top} = Q_{score} \cup Q_{pro}$.

- The set Q_{score}, which contains tuples with the top-*k* highest score, is found from lines 2 to line 6 . As seen in the ranking order on score of the probability data set, the first *k* tuples $\{t_1..t_k\}$ have the best-*k* score which is added into the answer set Q_{score} (line 4), and their top-*k* probabilities are equal to their probabilities as proven in Theorem 3 (line 3). Then, *bestPr* is initialised with the minimum top-*k* probabilities from t_1 to t_k (line 5) and the first value of p_{prev} is assigned the probability of the tuple having minimum pr^k_{top} which equals the *bestPr* (line 6).

- The set Q_{pro} containing non-dominated tuples is selected from lines 8 to 13. A current tuple t_i is determined to be dominated or not as follows. Firstly, the top-*k* probability must be calculated with generation rules $\mathcal{R}$ (line 9). These complex computation formulas are formally presented in section 2. After this, the top-*k* probability of the current tuple is compared with *bestPr*. If the top-*k* probability is greater than *bestPr*, the current tuple is a non-dominated tuple (line 10). Then, it is added into the answer set Q^k_{top} (line 12). The new values of *bestPr* and p_{prev} are next considered in lines 11 and 13.

- For effectiveness, the pruning rules and stopping rule are applied at lines 8 and 14, respectively. The first condition $(p(t_i) > bestPr)$ is presented for Theorem 4, and the second condition $(p(t_i) > p_{prev})$ is presented for Theorem 5 (line 8). Theorems 4 and 5 are applied to directly prune the current tuple without the need to calculate top-*k* probability. Therefore, the algorithm does not calculate the top-*k* probability, and jumps to the next tuple being considered. In addition, the stopping condition on the top-*k* best probability process is activated as outlined in Theorem 6 (line 14), which means there are no more non-dominated tuples in the rest of the tuples.

- Overall, the top-*k* best probability algorithm returns the answer $Q^k_{top} = Q_{score} \cup Q_{pro}$.

4. Semantics of top-*k* best probability queries and other top-*k* queries

This section first discusses the previous semantic ranking properties proposed in papers [3, 4] and proposes the new property "*k-best ranking scores*" based on users' expectations. Secondly, we examine which properties our approach and previous approaches possess.

4.1. *Semantics of ranking properties*

On probabilistic data, ranking queries have been studied in relation to their semantics. As the semantics of possible worlds is based on the relationship between probability and score, the answers to the ranking query have to be discussed to meet the static and dynamic postulates. The previous work [3, 4] formally defined the key properties of ranking query semantics on probabilistic data, based on the users' natural expectations. Five properties are presented and serve as benchmarks to evaluate and categorize the different semantics of ranking queries.

Let set $Q^k_{top}(\mathcal{D})$ be the answer to the top-k query on probabilistic data $\mathcal{D}$:

The first property is "*Exact-k*" being the users' natural expectations, which has k tuples in the answer set.

• *Exact-k*: If $|\mathcal{D}| \geq k$ then $|Q^k_{top}(\mathcal{D})| = k$. When $|Q^k_{top}(\mathcal{D})| \geq k$, it is weakly satisfied.

The second property is "*Containment*" for capturing the intuition of users on results returned.

• *Containment*: For any k, the set $Q^k_{top}(\mathcal{D}) \subset Q^{k+1}_{top}(\mathcal{D})$. When replacing $\subset$ with $\subseteq$, it is weakly satisfied.

The third property illustrates that the ranking answer should be unique.

• *Unique ranking*: If probabilistic data $\mathcal{D} \neq \mathcal{D}'$, then the result $Q^k_{top}(\mathcal{D}) \neq Q^k_{top}(\mathcal{D}')$.

The fourth property stipulates the limitation of score values on probabilistic databases.

• *Value invariance*: Let $\mathcal{D}$ be score set $(t_1.score \leq t_2.score \leq ... \leq t_n.score)$, which is replaced by another score set $(t_1.score' \leq t_2.score' \leq ... \leq t_n.score')$ namely $\mathcal{D}_r$. For any k, the set $Q^k_{top}(\mathcal{D}) = Q^k_{top}(\mathcal{D}_r)$.

The fifth property shows the dynamic postulate of ranking queries.

• *Stability*: Given a tuple $t_i = (v_i, p(t_i)) \in \mathcal{D}$, $\mathcal{D}'$ is a new probabilistic data set where t_i is replaced by following $t^{\uparrow}_i = (v^{\uparrow}_i, p^{\uparrow}(t_i))$ where $v^{\uparrow}_i$ is better than v_i, and $p^{\uparrow}(t_i) \geq p(t_i)$. If $t_i \in Q^k_{top}(\mathcal{D})$, then $t^{\uparrow}_i \in Q^k_{top}(\mathcal{D}')$.

However, it is desirable that users always expect the results of top-k queries to contain the k-best ranking score tuples, because the traditional top-k query always returns the set of k tuples which have the highest scores. Therefore, the extension of top-k queries to the probabilistic situation must return the answer set which includes the top-k tuples with the best ranking scores in order to satisfy the inheritance property. We propose the capability of providing these k-best scoring tuples as a property of top-k query semantics on probabilistic data. We name the sixth property as "*k-best ranking scores*".

• *k-best ranking scores*: Let probabilistic data $\mathcal{D} = (D_c, p)$, where D_c is the database without the probability dimension, and p is the set of probabilities. For any k, $Q^k_{top}(D_c) \subset Q^k_{top}(\mathcal{D})$.

In the next section, all the above six properties are used to analyse and discuss the semantics which satisfy the top-k queries on probabilistic databases between our approach and previous approaches.

4.2. *Top-k queries satisfying semantic properties*

We now investigate whether the existing top-k methods and our approach satisfy the semantic properties or not.

Method	*k*-best ranking score	Exact-*k*	Containment	Unique ranking	Value invariance	Stability
U-top-*k* [14]	Fail	Fail	Fail	Satisfied	Satisfied	Satisfied
U-*k*-ranks [14, 31]	Fail	Satisfied	Satisfied	Fail	Satisfied	Fail
Global-top-k [4]	Fail	Satisfied	Fail	Satisfied	Satisfied	Satisfied
E-Score [3]	Fail	Satisfied	Satisfied	Satisfied	Fail	Satisfied
E-Rank [3]	Fail	Satisfied	Satisfied	Satisfied	Satisfied	Satisfied
PT-*k* [1, 2]	Weak	Fail	Weak	Satisfied	Satisfied	Satisfied
U-popk [16]	Fail	Satisfied	Satisfied	Satisfied	Satisfied	Satisfied
Top-*k* bestPr	Satisfied	Weak	Weak	Satisfied	Satisfied	Satisfied

Table 4. Summary of top-*k* methods on the semantics of ranking query properties

Overall, Table 4 illustrates that our proposed approach covers the six properties of semantic ranking while the previous studies all fail in at least one of these properties. Hence, the answers to the top-k best probability query is better and meet the expectation of the users when compared to other existing work.

$*$ U-top-k [14]: Uncertain-top-k approach returns a set of top-k tuples, namely, a tuple vector. This vector has the highest aggregated probabilities in all possible worlds. The k-tuples in the tuple vector appear restrictively together in the same possible worlds. The U-top-k answers satisfy the *Unique ranking, Value invariance*, and *Value invariance* properties of ranking query semantics. However, it could not satisfy the *Exact-k* attribute when applied to small probabilistic database $\mathcal{D}$. The tuple vector with the highest aggregated probabilities has tuples less than k. It also fails on the *Containment* property. The answer to U-top-k is not a subset of U-top-$(k+1)$ due to complete disjunction from the top-$(k+1)$ as point out in [3, 4]. Moreover, the U-top-k answers violate the *k-best ranking scores* property. Given the example of the probabilistic data in Table 1, the U-top-2 answer is the vector (t_3, t_4) which has the highest top-2 probability vector 0.2272 in the possible world space. The answer to the U-top-2 query does not contain the top-2 highest score tuples t_1 and t_2. As a results, the answer to the U-top-2 query failed to satisfy the *k-best ranking scores* property.

$*$ U-k-ranks [14, 31]: Uncertain-k-ranks query returned a set of tuples, each of which is the maximum aggregated probabilities of all possible worlds ranked from 1 to k, therefore, each tuple in U-k-ranks results is the highest aggregated probability in its rank in the possible world space. The U-k-ranks answers satisfy the *Exact-k, Containment*, and *Value invariance* properties on semantics. However, it fails on the *unique ranking, Stability* and *k-best ranking scores* properties. For an example of U-3-ranks on the probabilistic data of Table 1, the answer set is $\{t_1, t_3\}$ due to tuple t_3 having maximum probability with $k = 2$ and $k = 3$. U-3-ranks fails on *unique ranking*. Moreover, it violates the *Stability* property because increasing the score or probability of tuples could change the consequence of ranks in possible worlds. The fact that it fails the *Stability* property is also pointed out in [3, 4]. The answer returned by U-3-rank does not contain tuple t_2 the second highest scoring tuple, and it also fails on the *k-best ranking scores* property.

$*$ Global-top-k [4]: The answer to the Global-top-k queries is a set of k tuples with the k-highest probabilities being the top-k answers in the possible world space. Global-top-k satisfies properties *Exact-k, Unique ranking, Value invariance,* and *Stability*. Global-top-k fails the two properties of *Containment* and *k-best ranking scores*. Running a Global-top-1 query and Global-top-2 query on Table 1 returns answer sets $\{t_1\}$ and $\{t_3, t_5\}$, respectively. It fails on the *Containment* property due to set $\{t_1\}$ not being a subset of set $\{t_3, t_5\}$. Moreover, Global-top-2 answers may not contain tuples t_1 and t_2, the top-2 best scores, therefore, Global-top-k also violates the *k-best ranking scores* property.

$*$ E-score [3]: Expected-score is a simple approach to multiply score and probability together as an expected-value of each tuple, then the E-score query returns the set of k-tuples having the highest expected-value. It is clear to see that the E-score method satisfies the properties *Exact-k, Containment, Unique ranking,* and *Stability*. However, it fails on properties *Value invariance* and *k-best ranking scores*. The magnitude of score and probability is a limitation of the E-score method. The magnitude of the normal expected score is a tuple having the low top-k probability and a high score, giving it the highest expected score. If the score has been adjusted to be just greater than the next highest score, it will fall down the ranking. Therefore, it fails on the *Value invariance* property. Moreover, the answer to the E-score query with $k = 2$ in Table 1 is a set $\{t_3, t_1\}$ which does not contain tuple $\{t_2\}$. Hence, it fails on the *k-best ranking scores* property.

$*$ E-rank [3]: A new Expected-ranking is produced by the Expected-score to select the top-k ranking query for probabilistic data. The authors used the ranking formula to calculate the new expected score for their proposal to remove the magnitude of normal expected score limitations. This method overcomes the shortcomings of E-score and satisfies almost all of the properties *Exact-k, Containment, Unique ranking, Value invariance,* and *Stability* as proven in paper [3]. However, it still fails on the *k-best ranking scores* property. Applying E-rank on the probabilistic data in Table 1, the final expected score is ranked $(t_1, t_3, t_2, t_5, t_4, t_6)$. The E-rank top-2 query answer is (t_1, t_3) which does not contain the top-2 highest score tuple t_2. Hence, the E-rank query answer fails to reach the *k-best ranking scores* property.

$*$ PT-k [1, 2]: M. Hue and J. Pei proposed the PT-k queries on probabilistic data using a user-specified probability threshold to cut off irrelevant candidate tuples having lower top-k probability. PT-k satisfies three properties *Unique ranking, Value invariance,* and *Stability*, as pointed out in paper [3]. Based on the description of the *containment* property, the PT-k method weakly satisfies this property since the same answer set can be returned from different k {k, k+1} with a fixed threshold. For instance, in Table 1, when we set the threshold to 0.4 for PT-2 and PT-3, the answer is the same set t_3. This also violates the *exact-k* property. The tuples in the answer are based on the users' specified probability threshold. The PT-k method is greatly affected by the result threshold. This was analysed in the introduction. Furthermore, the answer to the PT-2 query also does not contain the top-2 highest score tuples t_1, t_2. However, if the threshold is assigned a value less than or equal to 0.29, the PT-2 query answer $\{t_1, t_2, t_3, t_4, t_5\}$ contains the top-2 highest score tuples $\{t_1, t_2\}$. Therefore, the *k-best ranking scores'* property is weakly satisfied.

$*$ U-popk [16]: U-popk approach selects k tuples in order, in which, tuple $top - (i + 1)^{th}$ is a U-1-ranks after removing the $top - i$ tuples. The U-popk approach satisfies the *Exact-k, Unique ranking, Value invariance, Stability* ,and *Containment* properties of semantic ranking. However, U-popk violates the *k-best ranking scores* property. The U-popk query returns $\{t_1, t_3\}$ which excludes tuples t_2 from the top-2 highest score for the probabilistic data in Table 1.

$*$ Top-k bestPr: To analyse these properties for our top-k best probability queries, the four properties *Containment*, *Unique ranking*, *Value invariance*, and *Stability* proved similar to the PT-k method in paper [3]. The *Exact-k* property is defined as "the top-k answer that contains exactly k tuples". The top-k best probability query is *fairly satisfied* with this property, because the top-k best probability queries always return at *least k tuples as the answer*, which is a set of the top-k highest score tuples (Q_{score}). The answer to the top-k best probability query contains the top-k highest ranked score tuples Q_{score} and non-dominated tuples Q_{pro}. In the worst case, Q_{pro} can contain only one tuple t_{pmin} in Q_{score}. Hence, the *Exact-k* property also is reasonably acceptable in the top-k best probability approach. The last property *k-best ranking score* is also obtained by the top-k best probability result because it contains Q_{score}, the k-highest ranking tuples.

5. Experimental study

In this section, we report an extensive empirical study over a real and synthetic data set to examine our proposed approach which is more effective and efficient than the PT-k [1, 2]. All the experiments were conducted on a PC with a 2.33GHz Intel Core 2 Duo, 3 GB RAM, and 350GB HDD, running Windows XP Professional Operating system. An algorithm were implemented in C++.

5.1. Real data

We use the *International Ice Patrol Iceberg Sighting database* for our real data set[1]. This data set was used in previous work on ranking and skyline queries in probabilistic databases [1, 19, 34]. The IIP's mission is to survey, plot and predict iceberg drift to prevent icebergs threatening ships. The information in the database is iceberg activity in the North Atlantic in the area of 40 to 65 degrees north latitude and 39 to 57 degrees west longitude. The database in the IIP collection is structured in ASCII text files. The database contains a set of iceberg sightings represented by tuples, each tuple (t_i) including the important attributes (Sighting Source, Position, Date/Time, Iceberg Description, Drifted days). This information is very important to detect icebergs to prevent them threatening ships. Conducting top-k queries using the number of days of iceberg drift as a ranking score is considered an important tool for predicting icebergs' travel. Moreover, six types of observation equipments are used as sighting sources: R/V (Radar and Visual), VIS (VISual only), RAD (RADar only), SAT-LOW (LOW earth orbit SATellite), SAT-MED (MEDium earth orbit SATellite), and SAT-HIGH (HIGH earth orbit SATellite). Equipments used to observe icebergs can be unreliable due to the unstable environment or bad weather, therefore, erroneous information may be collected. These sighting sources are classified by differences in confidence levels as probabilities, which are R/V(0.7)$p(t_{i_1})$, VIS(0.6)$p(t_{i_2})$, RAD(0.5)$p(t_{i_3})$, SAT-LOW(0.4)$p(t_{i_4})$, SAT-MED (0.3)$p(t_{i_5})$, and SAT-HIGH(0.2)$p(t_{i_6})$. These numbers are also considered as probabilities of independent tuples in IIP data.

In the IIP database, some tuples from different sighting sources could detect the same iceberg at the same time. In this situation, based on latitude and longitude, we calculated the distance of between two tuples (icebergs). If the distance is less than 0.1 mile, they are considered to be two tuples of the same iceberg collected from different sources. Therefore, these tuples are mutually exclusive, as only one tuple can be correct. These tuples are recorded on the IIP probabilistic database which is controlled by exclusive rules $R_r = t_{r_1} \oplus t_{r_2} \oplus ... \oplus t_{r_q}$, and probabilities of these tuples are adjusted by the following formula:

[1]ftp://sidads.colorado.edu/pub/DATASETS/NOAA/G00807

$$p(t_{r_j}) = \frac{p(t_{r_j})}{\sum_{i=1}^{m} p(t_{r_i})} \times max(p(t_{r_1}), ..., p(t_{r_m}))$$

where $p(t_{r_j})$ is the probability of sighting sources of tuple t_{r_j}.

After reprocessing all tuples in the IIP database 2009, the IIP probabilistic database $\mathcal{D} = \{t_1, t_2, ..t_n\}$ contains 13,039 tuples and 2,137 exclusive rules. One of the exclusive rules is $R_r = t_7 \oplus t_8 \oplus t_9 \oplus t_{10}$. The proposed algorithm is executed on this IIP probabilistic database for the top-10 best probability query, the answer to this query being $(t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11})$ as shown in Table 5. Tuple t_1 has the highest value in the *Drifted Days* attribute, t_2 is the other tuple which has the same value or the second highest value in the *Drifted Days* attribute, and so on. The answer to the top-10 best probability query contains the 10 highest scoring tuples in the data set $(t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10})$. Tuple t_{11} is in the answer due to the fact that it has a top-10 probability which is better than a tuple with a minimum top-*k* probability in $(t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10})$. t_{11} is a non-dominated tuple. Moreover, tuples $(t_{12}, t_{13}, t_{14}, t_{15}, \ldots)$ are not in the answer because they are dominated by t_{11} on both ranking score and top-*k* probability.

Tuple	Drifted days	Pro. of tuples	Top-10 pro.
t_1	500.0	0.2917	0.2917
t_2	500.0	0.2333	0.2333
t_3	495.8	0.7	0.7
t_4	488.7	0.35	0.35
t_5	455.5	0.6	0.6
t_6	439.5	0.7	0.7
t_7	435.2	0.15	0.15
t_8	431.6	0.15	0.15
t_9	431.0	0.15	0.15
t_{10}	430.9	0.15	0.15
t_{11}	427.6	0.7	0.7
t_{12}	423.5	0.7	0.7
t_{13}	416.2	0.6	0.7
t_{14}	414.5	0.3	0.299
t_{15}	408.8	0.2	0.198
t_{16}	408.2	0.2	0.198

Table 5. highest scores of tuples in IIP (2009)

On this IIP probabilistic database, we also applied PT-10 query [1] by setting different thresholds, as shown in Figure 1. This illustrates that the number of tuples in the answers to PT-10 are dependent on the threshold. Firstly, the answers to the PT-10 query are required to contain the 10 best ranking score tuples. For this requirement, the threshold has to be set to less than or equal to the minimum value of the top-10 probability from t_1 to t_{10} in Table 5 which is less than or equal to 0.15. In this range, the number of tuples in the answer set of the PT-10 query is greater than 21, and these sets always contain tuples $\{t_{12}, t_{13}, t_{14}\}$ due to their top-10 probabilities being greater than 0.15, hence these tuples are redundant tuples as explained in the introduction. Moreover, if the threshold which is assigned from ranges greater than 0.15 to less than or equal to 0.7, the number of tuples in the PT-10 answer set are from 0 to 21. These results lost some important tuples in the 10 highest ranking score tuples, which are tuples $\{t_7, t_8, t_9, t_{10}\}$ because their probabilities are equal to 0.15. Lastly, the answer set of the PT-10 query is empty if users assigned a threshold value greater than 0.7. This is meaningless for the PT-10 query. Therefore, threshold plays a crucial role in selecting answers to the PT-*k* query. It is not easy for the users to choose a threshold to obtain suitable answers to PT-*k* queries. In our proposal, the users do not need to choose the threshold. The best answer of top-*k* queries will be returned without losing any important tuples or containing any redundant tuples in terms of the *k* best ranking score tuples and the non-dominated sets, whereupon the users can receive the best answers to the top-*k* best probability queries in both ranking score and top-*k* probability.

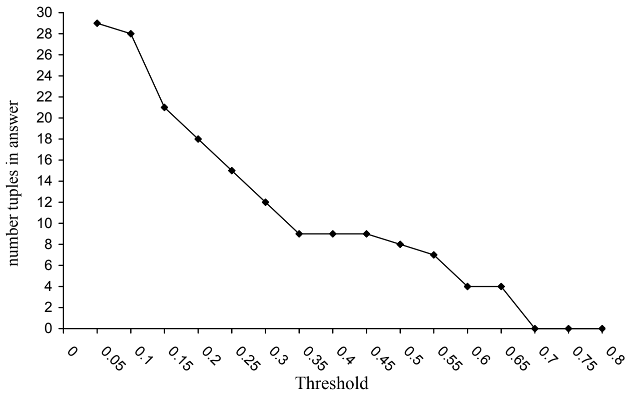

Figure 1. the answer to the PT-10 vs. thresholds.

We now compare the PT-*k* algorithm (PT-*k*) and our proposed algorithm (top-*k* best probability), based on the top-*k* traditional algorithm (top-*k* normal) to evaluate which is the most effective calculation and the most efficient answers. The top-*k* traditional algorithm simply returns the *k*-highest ranking scores in the data set. The top-*k* traditional algorithm has been widely used on certain databases, but has not been used on probabilistic databases. We mention this method as the axis for comparison of the PT-*k* results and top-*k* best probability results. For the PT-*k* algorithm, we assigned a minimum probability threshold of 0.15 to tuples t_1 to t_{15} of Table 5. This value was selected due to the fact that the PT-*k* has to satisfy the "*k-best ranking score*" attribute of Table 4 for comparison. It means that all answers to the PT-*k* will contain the top-*k* highest scoring tuples when *k* is run from 1 to 15. For these settings, we execute programs to obtain the results shown in Figures 2 and 3.

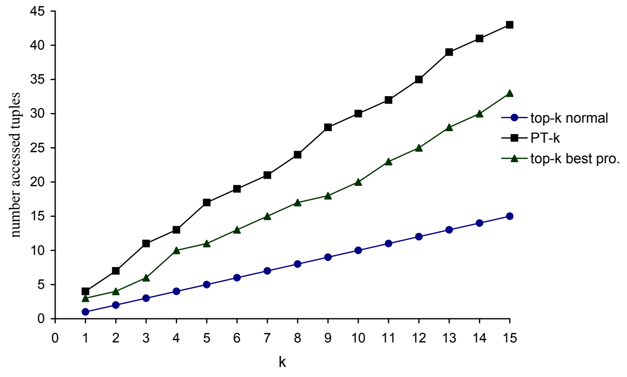

Figure 2. Accessed tuples vs. k

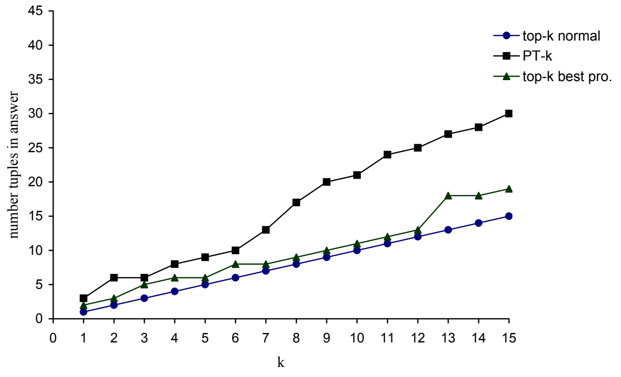

Figure 3. Tuples in answer vs. k

The effectiveness of the top-*k* best probability algorithm can be verified by counting the number of tuples which are accessed during the algorithm execution. The lower the number of tuples accessed, the more effective the algorithm. Figure 2 shows that the top-*k* best algorithm accesses fewer tuples than the PT-*k* algorithm for all *k* value. The top-*k* normal algorithm has the best performance in accessing the number of tuples. However, this algorithm has only been executed on certain data. Figure 3 shows the number of tuples in the answers to the PT-*k* queries, the top-*k* best probability queries, and the normal queries. The users always expect that the answers to the top-*k* queries on probabilistic data are concise. Figure 3 shows that all the answers to the top-*k* best probability queries contain less tuples than the answers to PT-*k* in all *k*. Also, they are closer to the top-*k* normal answers. This can explain why the PT-*k* answers can contain redundant tuples which are not non-dominated tuples, and the answers to top-*k* best probability queries have less tuples than PT-*k*, which is more meaning for users in terms of both ranking score and top-*k* probability. Hence, the answers to the top-*k* best probability queries are more efficient and concise than the answers to PT-*k* queries.

The results of the real data set clearly show that the top-*k* best probability algorithm removes the difficult of setting threshold value and reduces the number of accessed tuples compared to the PT-*k* algorithm. This makes the top-*k* best probability algorithm more effective. Moreover, the answers to the top-*k* best probability queries are concise and more efficient.

5.2. Synthetic data

In this section, we generated six data sets, which contain 5,000 tuples and have 500 generation rules. Various synthetic data sets are generated following normal distribution. Each generation rule contains the normal distribution N(10, 2). Probabilities of tuples in the six data sets are generated following normal distribution N(0.5, 0.0), N(0.6, 0.1), N(0.7, 0.2), N(0.8, 0.3), N(0.9, 0.4), and N(1, 0.5). We use these data sets to provide a deeper analysis of the comparative effectiveness and efficiency of the PT-*k* approach and our approach. We leave the threshold value unchanged at 0.5 which is a mean of range (0,1]. Then, the PT-50 algorithm and the top-50 best probability algorithm are executed on these generated probabilistic data sets. We obtained the results shown in Figure 4 and Figure 5.

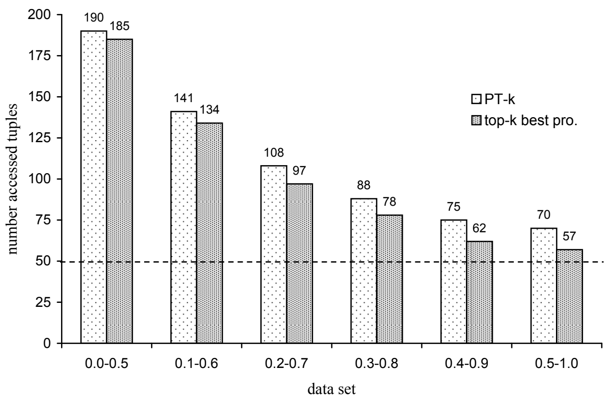

Figure 4. Accessed tuples vs. data sets

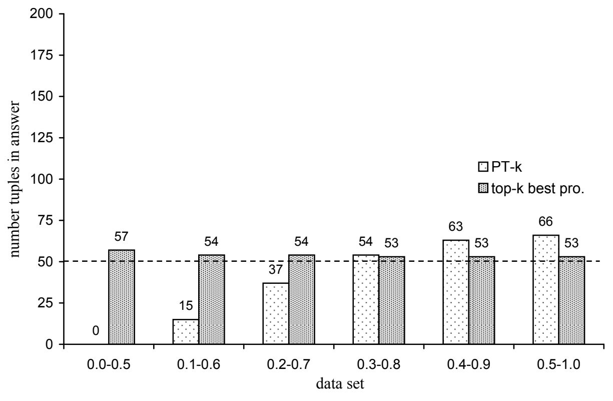

Figure 5. Tuples in answer vs. data sets

Figure 4 shows the number of tuples accessed by the PT-50 algorithm and the top-50 best probability algorithm in all generated data sets. The accessed tuples are the tuples accessed while computing the results of the query, even if the tuples are not included in the result set. Figure 4 shows that the top-50 best probability algorithm accessed less tuples than the PT-50 algorithm in all the different probabilistic data sets. Figure 5 illustrates the number of tuples in the results returned by the PT-50 algorithm and the top-50 best probability algorithm in six generated data sets. The number of tuples in answers to the top-50 best probability queries are more stable than the tuples in the answers to the PT-50 algorithm. The number of tuples in the answers to the top-50 best probability queries is from 53 to 57 while the number of tuples in the answers to the PT-50 queries is from 0 to 67. On data set (0.0, 0.5), the answer to PT-50 query is empty. This answer can be meaningless to users. In addition, the number of tuples in the PT-50 queries result on data sets (0.1,0.6) and (0.2,0.7) is 15 and 37 less than 50. These results are clearly losing some important answers of top-50 best ranking tuples. We choose to fix the value for the threshold of the PT-50 query, so the number of tuples in the result is not stable. Therefore, it would be very difficult for the users to set the right threshold value because different probability ranges produce a vastly different number of tuples in the results. In columns (0.4, 0.9), (0.5 1.0), the PT-50 results contain more tuples than the top-50 best probability results, because some tuples in the answer to the PT-50 can be redundant tuples, as previously discussed. Thirdly, when we combine Figures 4 and 5, the top-50 best probability approach is more effective than the PT-50 approach, the reason being the difference between the number of answer tuples and access tuples is smaller for the top-*k* best probability algorithm than PT-*k* for all probabilistic data sets.

For the synthetic probabilistic data, the experiment verified that the top-*k* best probability approach is more effective and efficient when compared to the PT-*k* for all probability ranges tested. The top-*k* best probability covers almost all the limitations of PT-*k*.

6. Conclusions

In this paper, we proposed a novel top-*k* best probability query for probabilistic databases, which selects the top-*k* best ranking score and the non-dominated tuples for users. The proposed method also satisfied the semantic ranking properties, which was better than other approaches in terms of semantic ranking queries. Firstly, several concepts and theorems from previous studies were formally defined and discussed in relation to the probabilistic database model. Secondly, we created a new definition for the top-*k* best probability query based on the top-*k* best ranking score and dominating concept. The algorithm was built with some proven pruning rules. Then, five semantic ranking properties and a new "k-best ranking scores" property were introduced to describe the semantic answer to top-*k* best probability queries. This proposed approach is an improvement on the previous state-of-the-art algorithms. The answers to the top-*k* best probability queries not only contain the top-*k* best score but also the top-*k* best probability tuples. Finally, the experimental results verified the efficiency and effectiveness of our approach on both real and synthetic data sets. The proposed approach has been shown to outperform the algorithm designed for the PT-*k* query in both efficiency and effectiveness.

In many real life domains, uncertain data is inherent in many applications and modern equipment. Therefore, discovering semantic answers to queries is a critical issue in relation to uncertain data. The proposed approach can be applied to modelling, managing, mining, and cleaning uncertain data.

References

[1] M. Hua, J. Pei, W. Zhang, X. Lin, Ranking queries on uncertain data: a probabilistic threshold approach, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, SIGMOD '08, ACM, New York, NY, USA, 2008, pp. 673–686.

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

[3] G. Cormode, F. Li, K. Yi, Semantics of ranking queries for probabilistic data and expected ranks, in: Proceedings of the 2009 IEEE International Conference on Data Engineering, ICDE '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 305–316.

[4] X. Zhang, J. Chomicki, Semantics and evaluation of top-k queries in probabilistic databases, Distrib. Parallel Databases 26 (1) (2009) 67–126.

[5] R. Cheng, D. V. Kalashnikov, S. Prabhakar, Evaluating probabilistic queries over imprecise data, in: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, SIGMOD '03, ACM, New York, NY, USA, 2003, pp. 551–562.

[6] I. F. Ilyas, G. Beskales, M. A. Soliman, A survey of top-k query processing techniques in relational database systems, ACM Comput. Surv. 40 (4) (2008) 11:1–11:58.

[7] A. D. Sarma, O. Benjelloun, A. Halevy, J. Widom, Working models for uncertain data, in: Proceedings of the 22nd International Conference on Data Engineering, ICDE '06, IEEE Computer Society, Washington, DC, USA, 2006.

[8] L. Getoor, Learning probabilistic relational models, Abstraction, Reformulation, and Approximation 1864 (2000) 322–323.

[9] W. Zhang, X. Lin, J. Pei, Y. Zhang, Managing uncertain data: probabilistic approaches, in: Web-Age Information Management, 2008.

[10] C. Aggarwal, P. Yu, A survey of uncertain data algorithms and applications, Knowledge and Data Engineering, IEEE Transactions on 21 (5) (2009) 609 –623.

[11] T. Pang-Ning, S. Michael, K. Vipin, Introduction to data mining, Library of Congress, 2006.

[12] S. McClean, B. Scotney, P. Morrow, K. Greer, Knowledge discovery by probabilistic clustering of distributed databases, Data Knowl. Eng. 54 (2) (2005) 189–210.

[13] R. Ross, V. S. Subrahmanian, J. Grant, Aggregate operators in probabilistic databases, J. ACM 52 (1) (2005) 54–101.

[14] M. A. Soliman, I. F. Ilyas, K. C.-C. Chang, Top-k query processing in uncertain databases, in: ICDE, 2007, pp. 896–905.

[15] S. Abiteboul, P. Kanellakis, G. Grahne, On the representation and querying of sets of possible worlds, SIGMOD Rec. 16 (3) (1987) 34–48.

[16] D. Yan, W. Ng, Robust ranking of uncertain data, in: Proceedings of the 16th international conference on Database systems for advanced applications - Volume Part I, DASFAA'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 254–268.

[17] T. M. N. Le, J. Cao, Top-k best probability queries on probabilistic data, in: Proceedings of the 17th international conference on Database systems for advanced applications - Volume Part II, DASFAA'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 1–16.

[18] T. Ge, S. Zdonik, S. Madden, Top-k queries on uncertain data: on score distribution and typical answers, in: Proceedings of the 35th SIGMOD international conference on Management of data, SIGMOD '09, ACM, New York, NY, USA, 2009, pp. 375–388.

[19] J. Li, B. Saha, A. Deshpande, A unified approach to ranking in probabilistic databases, The VLDB Journal 20 (2) (2011) 249–275.

[20] J. Pei, M. Hua, Y. Tao, X. Lin, Query answering techniques on uncertain and probabilistic data: tutorial summary, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, SIGMOD '08, ACM, New York, NY, USA, 2008, pp. 1357–1364.

[21] S. Zhang, C. Zhang, A probabilistic data model and its semantics, Journal of Research and Practice in Information Technology 35 (2003) 237–256.

[22] M. J. Atallah, Y. Qi, Computing all skyline probabilities for uncertain data, in: Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, PODS '09, ACM, New York, NY, USA, 2009, pp. 279–287.

[23] B. Qin, Y. Xia, Generating efficient safe query plans for probabilistic databases, Data Knowl. Eng. 67 (3) (2008) 485–503.

[24] K. Lange, Numerical analysis for statisticians, Springer, 1999.

[25] D. Papadias, Y. Tao, G. Fu, B. Seeger, An optimal and progressive algorithm for skyline queries, in: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, SIGMOD '03, ACM, New York, NY, USA, 2003, pp. 467–478.

[26] X. Lian, L. Chen, Shooting top-k stars in uncertain databases, The VLDB Journal 20 (6) (2011) 819–840.

[27] J. Pei, B. Jiang, X. Lin, Y. Yuan, Probabilistic skylines on uncertain data, in: Proceedings of the 33rd international conference on Very large data bases, VLDB '07, VLDB Endowment, 2007, pp. 15–26.

[28] S. Börzsönyi, D. Kossmann, K. Stocker, The skyline operator, in: Proceedings of the 17th International Conference on Data Engineering, IEEE Computer Society, Washington, DC, USA, 2001, pp. 421–430.

[29] J. Chomicki, P. Godfrey, J. Gryz, D. Liang, Skyline with presorting: Theory and optimizations., in: M. A. Klopotek, S. T. Wierzchon, K. Trojanowski (Eds.), Intelligent Information Systems, Advances in Soft Computing, Springer, 2005, pp. 595–604.

[30] A. Vlachou, M. Vazirgiannis, Ranking the sky: Discovering the importance of skyline points through subspace dominance relationships, Data Knowl. Eng. 69 (9) (2010) 943–964.

[31] X. Lian, L. Chen, Probabilistic ranked queries in uncertain databases, in: Proceedings of the 11th international conference on Extending database technology: Advances in database technology, EDBT '08, ACM, New York, NY, USA, 2008, pp. 511–522.

[32] M. A. Soliman, I. F. Ilyas, K. C.-C. Chang, Probabilistic top-k and ranking-aggregate queries, ACM Transaction Database System 33 (2008) 13:1–13:54.

[33] K. Yi, F. Li, G. Kollios, D. Srivastava, Efficient processing of top-k queries in uncertain databases with x-relations, IEEE Trans. on Knowl. and Data Eng. 20 (12) (2008) 1669–1682.

[34] C. Jin, K. Yi, L. Chen, J. X. Yu, X. Lin, Sliding-window top-k queries on uncertain streams, Proc. VLDB Endow. 1 (1) (2008) 301–312.