J Intell Inf Syst (2017) 48:215–242
DOI 10.1007/s10844-016-0403-x

Efficient pruning for top-K ranking queries on attribute-wise uncertain datasets

Jianwen Chen[1] · **Ling Feng**[2]

Received: 3 October 2013 / Revised: 7 March 2016 / Accepted: 21 March 2016/
Published online: 2 April 2016

Abstract Top-K ranking queries in uncertain databases aim to find the top-K tuples according to a ranking function. The interplay between score and uncertainty makes top-K ranking in uncertain databases an intriguing issue, leading to rich query semantics. Recently, a unified ranking framework based on parameterized ranking functions (PRFs) has been formulated, which generalizes many previously proposed ranking semantics. Under the PRFs based ranking framework, efficient pruning approach for Top-K ranking on datasets with tuple-wise uncertainty has been well studied in the literature. However, this cannot be applied to top-K ranking on datasets with attribute-wise uncertainty, which are often natural and useful in analyzing uncertain data in many applications. This paper aims to develop efficient pruning techniques for top-K ranking on datasets with attribute-wise uncertainty under the PRFs based ranking framework, which has not been well studied in the literature. We first develop a Tuple Insertion Based Algorithm for computing each tuple's PRF value, which reduce the time cost from the state of the art cubic order of magnitude to quadratic order of magnitude. Based on the Tuple Insertion Based Algorithm, three pruning strategies are developed to further reduce the time cost. The mathematics of deriving the Tuple Insertion Based Algorithm and corresponding pruning strategies are also presented. At last, we show that our pruning algorithms can also be applied to the computation of the top-k aggregate queries. The experimental results on both real and synthetic data demonstrate the effectiveness and efficiency of the proposed pruning techniques.

Keywords Pruning · Top-K ranking query · Uncertain dataset

✉ Jianwen Chen
chenjianwen@hit.edu.cn

Ling Feng
fengling@tsinghua.edu.cn

[1] School of Computer Science & Technology, Harbin Institute of Technology, Harbin, China

[2] Department of Computer Science & Technology, Tsinghua University, Beijing, China

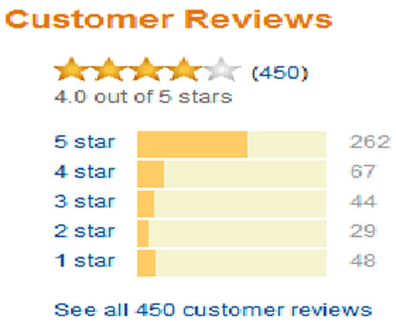
1 Introduction

There has been a great deal of research interest in uncertain databases (Sarma et al. 2006; Pei et al. 2008; Aggarwal and Yu 2009) due to the large amount of uncertain data emerging from a variety of application domains, such as information extraction and integration, market prediction, data mining and machine learning, and sensor network. A line of research work has focused on ranking uncertain data, with different rank semantics proposed in the literature. Recently, a unified approach to top-K ranking in uncertain databases, based on parameterized ranking functions (PRFs), has been formulated (Li et al. 2009), which generalizes many of the previously proposed ranking functions. The PRFs based ranking framework first maps each tuple in the uncertain database to a value (called PRF value in the following), and then sorts the tuples according to their PRF values. Computing the PRF value for each tuple in the database under the tuple-wise uncertain data model needs a quadratic time cost, which is too expensive. Wang et al. (Wang et al. 2011) further proposed efficient pruning techniques to improve the performance, which mainly aims for the tuple-wise uncertain data model. The tuple-wise uncertain data model is widely adopted to represent object uncertainty, where each tuple is considered as a description of an object, and has an additional attribute describing the probability that the corresponding object exists.

Parallel to the tuple-wise uncertain data model, the attribute-wise uncertain data model is widely used to represent value uncertainty where the uncertain attributes of the corresponding object are described as probability distributions. The value uncertainty arises naturally from many practical applications such as sensor readings (Deshpande et al. 2004; Kanagal and Deshpande 2008), and spatial objects with fuzzy locations (Tao et al. 2005). Developing efficient pruning techniques for the unified PRFs based ranking framework on dataset with value uncertainty is the aim of this paper.

Next, we illustrate a movie rating example, where the value uncertainty naturally arises.

Example 1 In Amazon, each customer can rate a movie from 1 star to 5 star. Thus, each movie's rating can be described as a discrete probability distribution, with each probability computed as the proportion of the customers giving the corresponding rating. Figure 1a shows a rating for a movie in website http://www.amazon.com. Figure 1b illustrates a table M containing 3 tuples, each of which describes a movie with an uncertain rating. For example, the first tuple describes a movie whose rating can take 3 star with probability 0.9, and



Customer Reviews

(450)
4.0 out of 5 stars

5 star	262
4 star	67
3 star	44
2 star	29
1 star	48

See all 450 customer reviews

(a)

	Name	Score
t_1	Movie1	$\{\langle 3, 0.9\rangle, \langle 4, 0.1\rangle\}$
t_2	Movie2	$\{\langle 2, 0.6\rangle, \langle 5, 0.4\rangle\}$
t_3	Movie3	$\{\langle 1, 0.8\rangle, \langle 2, 0.1\rangle, \langle 3, 0.1\rangle\}$

(b)

Fig. 1 **a** Rating for a movie in website http://www.amazon.com; **b** An example table M with discrete value uncertainty

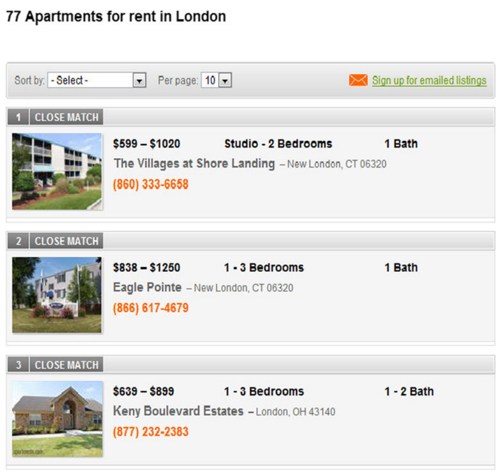
4 star with probability 0.1, meaning that 90 % of its rating customers give a 3 star, and 10 % of its rating customers give a 4 star. Note that we will continue to use this simple example for illustrating some basic computing methods in this paper, and in realistic applications there may be a large amount of movies each of which may have all the five possible ratings.

The PRFs based top-K ranking under the attribute-wise uncertain data model can be implemented by transforming the corresponding uncertain table into an equivalent uncertain table with x-tuple model, and then using the ranking techniques developed for the tuple-wise uncertain data model (Li et al. 2009). The corresponding algorithm is called Model Transforming Based Algorithm in the following. In this paper, we present a Tuple Insertion Based Algorithm for computing top-K ranking query on uncertain data with value uncertainty under the PRFs based ranking framework, where each tuple is considered as a whole unit instead of split tuples under the x-tuple model. Suppose that there are N tuples in the table under the attribute-wise model, and each tuple's score has an average number of M possible alternatives. In the Model Transforming Based Algorithm, the time cost of computing the PRF value for each tuple under the original attribute-wise model is $O(M^3N^2)$, while the time cost of the Tuple Insertion Based Algorithm is $O(MN^2)$. Based on the Tuple Insertion Based Algorithm, we further develop three efficient pruning strategies for computing the top-K ranking query, which avoid the computation of the exact PRF values of some tuples which are guaranteed not to be in the top-K list. We also show the mathematics of deriving the pruning algorithm. Our experimental results on both real data and synthetic data show orders of magnitude speedup over algorithms without pruning.

Despite the discrete value uncertainty, there are also many application domains where continuous value uncertainty naturally arises. For example, the rent of an automatically extracted apartment from the web may be missing and is assigned a range of possible prices $[low, upper]$ with the uniform distribution. In a sensor network, the measurement may be inaccurate due to noises and is usually presented as a continuous probability distribution on possible values. Typically, to handle such uncertain data, an uncertain table is used to store a set of tuples each of which comes with a score value represented as a continuous probability density function (pdf). Figure 2a shows the rents of some apartments from the website http://www.apartments.com, where the rent of each apartment is uncertain in an interval. Figure 2b illustrates a table with three tuples, where each tuple has a score described as a continuous probability density function (pdf). We also extend the Tuple Insertion Based Method and corresponding pruning algorithm for efficient ranking of uncertain data with continuous value uncertainty.

We further formulate the top-K aggregate query, which ranks groups of tuples by their aggregate values and returns the K groups with the highest aggregates, under the attribute-wise uncertain data model and the PRFs based ranking framework. This requires more time costs than top-K ranking query and is thus more challenging. Our experiments show that our pruning algorithm is also efficient for computing the top-K aggregate query.

This paper is organized as follows. In Section 2, we review the attribute-wise uncertain data model and the PRFs based ranking framework. In Section 3, we present an equivalent PRF value computing method, which forms the basis of our pruning algorithm. In Section 4, we present the mathematical derivation for our pruning techniques and corresponding algorithms for top-K queries. In Section 5, we formulate the top-K aggregate query under the attribute-wise uncertain data model and the PRFs based ranking framework, and apply our pruning techniques to it. Section 6 presents our experimental results. Section 7 reviews the related work and Section 8 concludes the paper.



77 Apartments for rent in London

Sort by: - Select - Per page: 10 Sign up for emailed listings

1 CLOSE MATCH

$599 – $1020 Studio - 2 Bedrooms 1 Bath
The Villages at Shore Landing – New London, CT 06320
(860) 333-6658

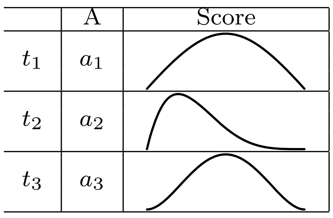
2 CLOSE MATCH

$838 – $1250 1 - 3 Bedrooms 1 Bath
Eagle Pointe – New London, CT 06320
(866) 617-4679

3 CLOSE MATCH

$639 – $899 1 - 3 Bedrooms 1 - 2 Bath
Keny Boulevard Estates – London, OH 43140
(877) 232-2383

(a)



(b)

Fig. 2 **a** Rents of apartments in website http://www.apartments.com; **b** An example table with continuous value uncertainty

2 Review of top-K ranking queries

In this section, we first review the top-K ranking queries in a deterministic database. Then, we review the attribute-wise uncertain data model for representing value uncertainty and the PRFs based ranking framework.

2.1 Top-K ranking query in a deterministic database

In a deterministic database, a top-K ranking query on a table T returns the K tuples with the largest scores, which can be defined as a function $F(R_1, R_2, \ldots, R_m)$ of attributes $R1$, $R2$, ..., R_m. A survey on the subject can be found in (Ilyas et al. 2008).

An example top-K query on the table T in a deterministic database (Fig. 3a) is as follows:

```
SELECT Name
FROM T
ORDER BY R1 + R2
LIMIT 2
```

According to the score $R1 + R2$, the top-2 tuples in the answer is $\{M3, M1\}$.

	Name	R1	R2
t_1	M1	2	3
t_2	M2	1	3
t_3	M3	3	4

(a)

	Name	R1	R2
t_1	M1	$\{\langle 3, 0.9\rangle, \langle 4, 0.1\rangle\}$	$\{\langle 1, 0.6\rangle, \langle 2, 0.4\rangle\}$
t_2	M2	$\{\langle 2, 0.6\rangle, \langle 5, 0.4\rangle\}$	$\{\langle 3, 0.2\rangle, \langle 4, 0.8\rangle\}$
t_3	M3	$\{\langle 1, 0.8\rangle, \langle 2, 0.2\rangle\}$	$\{\langle 1, 0.3\rangle, \langle 2, 0.7\rangle\}$

(b)

Fig. 3 **a** An example table T in a deterministic database **b** An example table T in a probabilistic database

2.2 Top-K ranking query under the attribute-wise uncertain data model

Without loss of generality, we assume the uncertain database contains simply one table T with N tuples $t_1, t_2, \ldots, t_N$. The commonly adopted attribute-wise uncertain data model is used to represent the value uncertainty. There can be multiple attributes in T whose value are random variables, described as **pmf**s (probability mass functions) or **pdf**s (probability density functions). Each tuple t_i in T has one score s_i, specified as a function $F(R_1, R_2, \ldots, R_m)$ of the tuple attributes $R_1, R_2, \ldots, R_m$. Science $R_1, R_2, \ldots, R_m$ are all random variables, $F(R_1, R_2, \ldots, R_m)$ is a random variable accordingly. The **pmf** or **pdf** of $F(R_1, R_2, \ldots, R_m)$ can be derived from the **pmf**s or **pdf**s of $R_1, R_2, \ldots, R_m$ (DeGroot and Schervish 2011). Figure 3b shows an example table T in a probabilistic database, where attributes R_1 and R_2 are described as **pmf**s. For tupe t_1, its attribute R_1 has value 3 with probability 0.9 and value 4 with probability 0.1, R_2 has value 1 with probability 0.6 and value 2 with probability 0.4. If the score used to rank the tuples is $R_1 + R_2$, then the score of t_1 has a **pmf** $\{\langle 4, 0.54\rangle, \langle 5, 0.42\rangle, \langle 6, 0.04\rangle\}$. In the discrete attribute-wise uncertain data model, s_i is viewed as a discrete random variable described as a **pmf** , $\{\langle s_{i1}, p_{i1}\rangle, \langle s_{i2}, p_{i2}\rangle, \ldots, \langle s_{in_i}, p_{in_i}\rangle\}$, meaning that s_i can take values $s_{i1}, s_{i2}, \ldots, s_{in_i}$ with probabilities $p_{i1}, p_{i2}, \ldots, p_{in_i}$. We also call $s_{i1}, s_{i2}, \ldots, s_{in_i}$ the n_i alternatives of score s_i. The different random variables corresponding to different tuple scores are assumed to be independent. A table following this model is illustrated in Fig. 4. In the continuous attribute-wise uncertain data model, the **pmf** used to describe score s_i is substituted for a **pdf** (probability density function), denoted as f_i.

In the PRFs based ranking framework, a parameterized ranking function (PRF), $\gamma : T \rightarrow R$ is defined to be

$$\gamma(t) = \sum_{i=1}^{N} \omega_i P(r(t) = i) \tag{1}$$

where t is a tuple in T, $r(t)$ is the rank of t, and $P(r(t) = i)$ is the probability that tuple t is ranked at the position i. $\omega_1, \omega_2, \ldots, \omega_N$ are monotonically non-increasing real numbers which weight the probabilities that t is ranked at the corresponding positions. Note that from the definition in (Li et al. 2009), the PRF function and all the weights ω_i can take complex numbers. In our work, we constrain the corresponding domains to real numbers, which represent the majority of applications.

For each tuple t, $\gamma(t)$ is called its PRF value. A top-K ranking query on table T first computes all its tuples' PRF values, and then returns the K tuples with the largest PRF values.

Note that the PRFs based ranking framework generalizes a variety of different ranking functions through appropriate weight parameter choices. For example, when $\omega_1 = 1, \omega_2 = 0, \ldots, \omega_N = 0$, all the tuples are ranked according to the probabilities that they have the

tuple	Score
t_1	$\{\langle s_{11}, p_{11}\rangle, \langle s_{12}, p_{12}\rangle, \ldots, \langle s_{1n_1}, p_{1n_1}\rangle\}$
t_2	$\{\langle s_{21}, p_{21}\rangle, \langle s_{22}, p_{22}\rangle, \ldots, \langle s_{2n_2}, p_{2n_2}\rangle\}$
...	...
t_N	$\{\langle s_{N1}, p_{N1}\rangle, \langle s_{N2}, p_{N2}\rangle, \ldots, \langle s_{Nn_N}, p_{Nn_N}\rangle\}$

Fig. 4 A Table following the discrete attribute-wise uncertain data model

largest ratings; when $\omega_1 = 1, \ldots, \omega_K = 1, \omega_{K+1} = 0, \ldots, \omega_N = 0$, all the tuples are ranked according to the probabilities that they are ranked at the top-K places, this corresponds to the PT-K semantics proposed in (Hua et al. 2008); when $\omega_i = N - i (1 \leq i \leq N)$, the PRF values will produce the same ranking as the expected rank semantics (Jestes et al. 2011). Note that the expected score semantics, which ranks the tuples by their expected scores, cannot be simulated using the PRFs based ranking framework. As illustrated in (Jestes et al. 2011), the expected score semantics is usually rejected due to its sensitivity to the score values. In realistic applications, replacing expected score semantics with expected rank semantics has the effect of moderating the influence of outliers.

Under the discrete attribute-wise uncertain data model, the PRFs based ranking framework can be interpreted under the prevalent possible world semantics (Sarma et al. 2006). The set of all possible worlds is denoted by $PW = \{pw_1, pw_2, \ldots, pw_n\}$. Each possible world pw_i is formed by taking one alternative for each tuple's uncertain score and has a probability $P(pw_i)$ computed as the product of the corresponding alternatives' probabilities. Each possible world corresponds to a single deterministic database instance. For each tuple t, its rank in each possible world pw_i, denoted as $r_{pw_i}(t)$, can be determined according to all the alternative scores in pw_i. The probability that tuple t is ranked at the ith position can be computed as:

$$P(r(t) = i) = \sum_{pw \in PW \wedge r_{pw}(t) = i} P(pw) \tag{2}$$

Then, all the tuple's PRF values can be computed from (1), and the top-K tuples with the largest PRF values can be determined.

Note that for a tuple t and a possible world pw, if there are x tuples whose scores are larger than t's score in pw, we define the rank of t as $x + 1$. For example, suppose that there are 6 tuples in a possible world with scores 4, 3, 3, 2, 2, 1, then, their ranks are 1, 2, 2, 4, 4, 6 respectively.

Example 2 Figure 5 illustrates all the possible worlds for the uncertain table in Fig. 1b. Tuple t_1 is ranked in the first place in possible worlds pw_1, pw_2, pw_3, pw_7, pw_8, pw_9, since t_1 has the highest score in all the possible worlds according to the deterministic database ranking semantics. Hence, $P(r(t_1) = 1) = P(pw_1) + P(pw_2) + P(pw_3) + P(pw_7) + P(pw_8) + P(pw_9) = 0.6$. Similarly, we can compute $P(r(t_1) = 2) = 0.4$,

possible world	probability
$pw_1 : \{t_1 : 3, t_2 : 2, t_3 : 1\}$	0.9*0.6*0.8=0.432
$pw_2 : \{t_1 : 3, t_2 : 2, t_3 : 2\}$	0.9*0.6*0.1=0.054
$pw_3 : \{t_1 : 3, t_2 : 2, t_3 : 3\}$	0.9*0.6*0.1=0.054
$pw_4 : \{t_1 : 3, t_2 : 5, t_3 : 1\}$	0.9*0.4*0.8=0.288
$pw_5 : \{t_1 : 3, t_2 : 5, t_3 : 2\}$	0.9*0.4*0.1=0.036
$pw_6 : \{t_1 : 3, t_2 : 5, t_3 : 3\}$	0.9*0.4*0.1=0.036
$pw_7 : \{t_1 : 4, t_2 : 2, t_3 : 1\}$	0.1*0.6*0.8=0.048
$pw_8 : \{t_1 : 4, t_2 : 2, t_3 : 2\}$	0.1*0.6*0.1=0.006
$pw_9 : \{t_1 : 4, t_2 : 2, t_3 : 3\}$	0.1*0.6*0.1=0.006
$pw_{10} : \{t_1 : 4, t_2 : 5, t_3 : 1\}$	0.1*0.4*0.8=0.032
$pw_{11} : \{t_1 : 4, t_2 : 5, t_3 : 2\}$	0.1*0.4*0.1=0.004
$pw_{12} : \{t_1 : 4, t_2 : 5, t_3 : 3\}$	0.1*0.4*0.1=0.004

Fig. 5 All the possible worlds for the uncertain table M in Fig. 1b

$P(r(t_1) = 3) = 0.0$. If the weights are set as $\omega_1 = 1$, $\omega_2 = \frac{1}{2}$, $\omega_3 = \frac{1}{3}$, then, $\gamma(t_1) = \omega_1 * P(r(t_1) = 1) + \omega_2 * P(r(t_1) = 2) + \omega_3 * P(r(t_1) = 3) = 0.8$. Similarly, we can compute $\gamma(t_2) = 0.7$, $\gamma(t_3) = 0.39$. The top-2 tuples with the largest PRF values are t_1 and t_2.

Since there can be exponentially many combinations of tuples each generating a distinct possible world, the computing method of enumerating all the possible worlds is infeasible. In (Li et al. 2009), a Model Transforming Based Computing Method is proposed. To compute all the tuples' PRF values, T can be first transformed into an equivalent table T' with x-tuple model(Agrawal et al. 2006). In the x-tuple model, each table in a probabilistic database is composed of a number of x-tuples. Each x-tuple in turn consists of one or more alternatives with corresponding probabilities. In real applications, each x-tuple describes an uncertain object or an uncertain event, with the possible instances described as alternatives. All the x-tuples are assumed to be independent, and all the alternatives in each x-tuple are assumed to be exclusive. In table T, each tuple t_i with score $\{ \langle s_{i1}, p_{i1}\rangle, \langle s_{i2}, p_{i2}\rangle, \ldots, \langle s_{in_i}, p_{in_i}\rangle\}$ is transformed into n_i alternatives in T', corresponding to n_i tuples in T'. For each tuple t_i in T, a generation rule is added for T', representing the exclusive relation of the alternatives originating from t_i.

Figure 6 shows the table M' under the x-tuple model transformed from the table M under the attribute-wise model in Fig. 1b. The first tuple in M is transformed into two tuples in M', corresponding to the two possible ratings for Movie1, and a corresponding generation rule is produced to represent the mutual exclusive relation of the two tuples in M'. The corresponding tuples and generation rules in M' for the other two tuples in M are produced similarly.

To compute the PRF value for a tuple in the original table T, we can first compute the PRF values of its corresponding tuples in T' and adding them all. For each tuple in T', the time cost for computing its PRF value is quadratic to the number of tuples in T' (Li et al. 2009). Suppose that there are N tuples in the table under the attribute-wise model, and each tuple has an average number of M possible alternatives. In the Model Transforming Based Algorithm, the time cost of computing the PRF value for each tuple under the original attribute-wise model is $O(M^3N^2)$, since for each original tuple there are an average number of M tuples under the x-tuple model and computing the PRF value for each of them needs a time cost of $O((MN)^2)$.

We cannot apply the efficient pruning techniques under the x-tuple model as presented in (Wang et al. 2011) to the attribute-wise model, since the efficient pruning techniques under the x-tuple model aims to find the top-K tuples with the largest PRF values without computing the exact PRF values of a large portion of the tuples in the uncertain database,

Fig. 6 Table M' under the x-tuple model corresponding to Table M in Fig. 1b under the attribute-wise model. The generation rules are $t'_1 \oplus t'_2$, $t'_3 \oplus t'_4$, $t'_5 \oplus t'_6 \oplus t'_7$

	Name	Score	Prob.
t'_1	Movie1	3	0.9
t'_2	Movie1	4	0.1
t'_3	Movie2	2	0.6
t'_4	Movie2	5	0.4
t'_5	Movie3	1	0.8
t'_6	Movie3	2	0.1
t'_7	Movie3	3	0.1

while computing the PRF value of a tuple in a table under the attribute-wise uncertain data model needs all the PRF values of the corresponding tuples in the transformed table under the x-tuple model.

Under the continuous attribute-wise uncertain data model, in (Li and Deshpande 2010), a generating function based method is proposed to compute the PRF value for ranking uncertain tuples. This produces a cubic algorithm. The real line is partitioned into small intervals such that the pdf of each tuple can be approximated as a single polynomial in each small interval. For each small interval I_j, let M_j be the set of tuples whose pdf support (the support of pdf f is defined as $supp(f) = \{x|f(x) > 0\}$) contains I_j and $m_j = |M_j|$, the time cost of computing the PRF value for each tuple is $O(\sum_j m_j^3)$.

3 A tuple insertion based computing method for top-K queries

In this section, we present an equivalent Tuple Insertion Based Method for computing the PRF values, which forms the basis of our pruning algorithm.

For tuple t in T, to compute $\gamma(t)$, we start with a tuple set S which only contains t. Then, in S, t is ranked at the first place with probability 1.0, and the PRF value of t, when only the tuples in S is considered, is $\gamma^{(1)}(t) = \omega_1 P(r(t) = 1) = \omega_1$. Here, we use $\gamma^{(j)}(t)$ to denote the PRF value of t when only the j elements in S are considered. Then, a new tuple in T is added to S, and $\gamma^{(2)}(t)$ is computed. This process continues until all the N elements are added to S and $\gamma^{(N)}(t)$ is computed. At last, $\gamma(t) = \gamma^{(N)}(t)$. In the following, for convenience of description, we use t_1 to denote t, and t_i to denote the ith added tuple from T.

In the following, we first describe the Tuple Insertion Based Computing Method under the discrete attribute-wise uncertain data model.

Suppose that $t_1, t_2, \ldots, t_j$ have been added to S. We use $\rho_{uv}^{(j)}$ to denote, when only considering the j tuples in S, the probability that t_1's score takes value s_{1u} and t_1 is ranked at the vth position. When t_{j+1} is added to S, if t_{j+1}'s score is less than or equal to t_1's score, then t_1's ranking position will remain unchanged; otherwise, t_1's ranking position will increase by 1.

Hence, $\rho_{uv}^{(j+1)}$ can be computed as

$$\rho_{uv}^{(j+1)} = P(s_{j+1} \leq s_{1u})\rho_{uv}^{(j)} + P(s_{j+1} > s_{1u})\rho_{u(v-1)}^{(j)} \tag{3}$$

Combining the initial conditions that $\rho_{u1}^{(1)} = p_{1u}$, $\rho_{u0}^{(j)} = 0$, and $\rho_{u(j+1)}^{(j)} = 0$, $\rho_{uv}^{(N)}$ can be computed from (3) deductively.

Then, $P(r(t_1) = i)$ can be computed as

$$P(r(t_1) = i) = \sum_{u=1}^{n_1} \rho_{ui}^{(N)} \tag{4}$$

and $\gamma(t_1)$ can be computed from (1).

Example 3 We compute the PRF values for the tuples of the uncertain table in Fig. 1b by using the Tuple Insertion Based Computing Method. For tuple t_1, when only t_1 is added to S, we have $\rho_{11}^{(1)} = 0.9$, $\rho_{21}^{(1)} = 0.1$, meaning that, when considering the only one tuple in S, the probability that t_1 takes the value $s_{11} = 3$ and is ranked at the first place is 0.9 and the probability that t_1 takes the value $s_{12} = 4$ and is ranked at the first place is 0.1. After t_2 is added to S, when only considering the two tuples in S, we have

$$\begin{aligned}
\rho_{11}^{(2)} &= P(s_2 \le s_{11})\rho_{11}^{(1)} + P(s_2 > s_{11})\rho_{10}^{(1)} \\
&= 0.6 * 0.9 + 0.4 * 0 = 0.54 \\
\rho_{12}^{(2)} &= P(s_2 \le s_{11})\rho_{12}^{(1)} + P(s_2 > s_{11})\rho_{11}^{(1)} \\
&= 0.6 * 0 + 0.4 * 0.9 = 0.36 \\
\rho_{21}^{(2)} &= P(s_2 \le s_{12})\rho_{21}^{(1)} + P(s_2 > s_{12})\rho_{20}^{(1)} \\
&= 0.6 * 0.1 + 0.4 * 0 = 0.06 \\
\rho_{22}^{(2)} &= P(s_2 \le s_{12})\rho_{22}^{(1)} + P(s_2 > s_{12})\rho_{21}^{(1)} \\
&= 0.6 * 0 + 0.4 * 0.1 = 0.04
\end{aligned} \tag{5}$$

Similarly, after t_3 is added to S, according to (3), we have $\rho_{11}^{(3)}(t_1) = 0.54$, $\rho_{12}^{(3)}(t_1) = 0.36$, $\rho_{13}^{(3)}(t_1) = 0.0$, $\rho_{21}^{(3)}(t_1) = 0.06$, $\rho_{22}^{(3)}(t_1) = 0.04$, $\rho_{23}^{(3)}(t_1) = 0.0$. Finally,

$$\begin{aligned}
P(r(t_1) = 1) &= \rho_{11}^{(3)} + \rho_{21}^{(3)} = 0.6 \\
P(r(t_1) = 2) &= \rho_{12}^{(3)} + \rho_{22}^{(3)} = 0.4 \\
P(r(t_1) = 3) &= \rho_{13}^{(3)} + \rho_{23}^{(3)} = 0.0
\end{aligned} \tag{6}$$

and

$$\gamma(t_1) = \sum_{i=1}^{3} \omega_i P(r(t_1) = i) = 0.8 \tag{7}$$

if $\omega_1, \omega_2, \omega_3$ are set to $1, \frac{1}{2}, \frac{1}{3}$ respectively, as in Example 2. $\gamma(t_2) = 0.7$ and $\gamma(t_3) = 0.39$ can be computed similarly, and we get the same results as in Example 2.

The Tuple Insertion Based Computing Method conforms to the possible world semantics. We put the proof in Appendix A. For each tuple t whose score has M alternatives, the time cost of computing the PRF value for t is $O(MN^2)$, since there are M alternatives and for each alternative the time cost of computing $\rho_{uv}^{(N)}$ according to (3) is $O(N^2)$.

Next, we describe the Tuple Insertion Based Computing Method under the continuous attribute-wise uncertain data model. We use $\gamma(t_1)$ as an example to illustrate the process of computing the PRF values. $\gamma(t_1)$ can be computed as

$$\gamma(t_1) = \sum_{i=1}^{N} \omega_i P(r(t_1) = i) = \sum_{i=1}^{N} \omega_i \int_{-\infty}^{+\infty} \mathcal{F}_i^{(N)}(x)dx \tag{8}$$

where $\mathcal{F}_i^{(N)}(x)$ $(1 \le i \le N)$ can be computed recursively as follows:

$$\mathcal{F}_1^{(1)}(x) = f_1(x) \tag{9}$$

and

$$\begin{aligned}
\mathcal{F}_1^{(u+1)}(x) &= \mathcal{F}_1^{(u)}(x) \int_{-\infty}^{x} \rho_{u+1}(t)dt \\
\mathcal{F}_2^{(u+1)}(x) &= \mathcal{F}_1^{(u)}(x)(1 - \int_{-\infty}^{x} f_{u+1}(t)dt) + \mathcal{F}_2^{(u)}(x) \int_{-\infty}^{x} f_{u+1}(t)dt \\
&\dots \\
\mathcal{F}_u^{(u+1)}(x) &= \mathcal{F}_{u-1}^{(u)}(x)(1 - \int_{-\infty}^{x} f_{u+1}(t)dx + \mathcal{F}_u^{(u)}(x) \int_{-\infty}^{x} f_{u+1}(t)dt \\
\mathcal{F}_{u+1}^{(u+1)}(x) &= \mathcal{F}_u^{(u)}(x)(1 - \int_{-\infty}^{x} f_{u+1}(t)dt)
\end{aligned} \tag{10}$$

The differential argument to derive the (8), (9) and (10) is as follows. We start with a tuple set S only containing t_1. Then, $t_2, t_3, \ldots, t_N$ are added to S in order. After $t_1, t_2, \ldots, t_u$ are added to S, when only considering the u tuples in S, the probability that the score of t_1 lies in $[x, x + dx]$ and is ranked at the ith position is expressed as $\mathcal{F}_i^u(x)dx$. After t_{u+1} is added to S, when only considering the $u + 1$ groups in S, the probability that the score of t_1 lies in $[x, x + dx]$ and is ranked at the ith position, expressed as $\mathcal{F}_i^{u+1}(x)dx$, can be computed as the sum of the two parts: first, the probability that the score of t_1 lies in $[x, x+dx]$ and t_1 is ranked at the $i-1$th position before t_{u+1} is added to S and the new added tuple t_{u+1} is ranked before t_1 (equivalently, the score of t_{u+1} is larger than x); and second, the probability that the score of t_1 lies in $[x, x + dx]$ and t_1 is ranked at the ith position before t_{u+1} is added to S and the new added tuple t_{u+1} is ranked after t_1 (equivalently, the score of t_{u+1} is less than x). Since the probability that the score of t_{u+1} is larger than x can be computed as $1 - \int_{-\infty}^{x} f_{u+1}(t)dt$, and the probability that the score of t_{u+1} is less than x can be computed as $\int_{-\infty}^{x} f_{u+1}(t)dt$, we have

$$\mathcal{F}_i^{(u+1)}(x)dx = (\mathcal{F}_{i-1}^{(u)}(x)(1 - \int_{-\infty}^{x} f_{u+1}(t)dt) + \mathcal{F}_i^{(u)}(x)\int_{-\infty}^{x} f_{u+1}(t)dt)dx \tag{11}$$

For the cases when $i = 1$ and when $i = u+1$, we have $\mathcal{F}_0^{(u)}(x)dx = 0$ and $\mathcal{F}_{u+1}^{(u)}(x)dx = 0$, which ensure the first equation and the last equation in (14). The probability that t_1 is ranked at the ith position when only considering the $u+1$ tuples in S can be computed by integrating both sides of (10).

For any other tuple t_i in T, $\gamma(t_i)$ can be computed similarly by considering t_i as the first tuple, and the other tuples are inserted into S in an arbitrary order. The order of the other tuples does not affect the computing result for $\gamma(t_i)$. However, taking different orders may require different time costs. We shall describe the order of the tuples adopted in our pruning strategies in the next section.

In this paper, we assume that all the pdfs have bounded support. Hence, all the integrals in the above formulas defined with infinite integral limits can be transformed into equivalent integrals with finite limits. For example, suppose pdf f has a bounded support $[a, b]$, then $\int_{-\infty}^{+\infty} f(x)dx = \int_a^b f(x)dx$. If a random variable has a pdf with unbounded support, we truncate the distribution and ignore the tail with minuscule probability. Suppose X is a random variable with mean μ and variance σ^2, we have $P(|X - \mu| \geq t\sigma) \leq \frac{1}{t^2}$ which can be derived from the Chebychev's Inequality. Hence, by selecting appropriate t, a bounded dominating range of the support can be decided.

All the integrals are computed by the Simpson method. For a function $f : \mathcal{R} \to \mathcal{R}$ on an interval $[a, b]$, the integral of f can be computed as follows according to the Simpson method:

$$\int_a^b f(x)dx = \sum_{i=1}^{n} S_i(f) \tag{12}$$

where $[a, b]$ is divided into n subintervals with equal length δ, and in each subinterval $[x_i, x_{i+1}]$,

$$S_i(f) = \frac{x_{i+1} - x_i}{6}(f(x_i) + 4f(\frac{x_i + x_{i+1}}{2}) + f(x_{i+1})) \tag{13}$$

Suppose that M_{f_i} is the number of subintervals of the support of f_i divided by δ in the Simpson method, the time cost of computing $\gamma(t_i)$ by using the Tuple Insertion Based Method under the continuous attribute-wise uncertain data model is $O(M_{f_i} N^2)$, where N is the number of tuples.

4 Pruning for top-K queries

The basic algorithm for computing a top-K ranking query on table $T = \{t_1, t_2, \ldots, t_N\}$ is first computing $\gamma(t_1)$, $\gamma(t_2)$, ..., $\gamma(t_N)$, selecting the K largest, and then returning the corresponding tuples.

The skeleton of our pruning algorithm is first selecting K tuples, and computing their PRF values. For each other tuple t in T, we compute an upper bound for the PRF value of t. If the upper bound is less than the Kth largest PRF value retrieved so far, then, the computing of the exact PRF value for t is avoided and the corresponding time cost is saved. Otherwise, the exact PRF value for t is computed and the top K list is updated. The pruning algorithm is based on the Tuple Insertion Based Computing Method described in Section 3, and we call it Upper Bound Estimation Based Pruning Algorithm in the following.

The key point is to decide an upper bound for t's PRF value, without computing its exact value. The main idea comes from the following three strategies.

Strategy 1 As described in the Tuple Insertion Based Computing Method, we also use t_1 to denote the new tuple from T. In the Tuple Insertion Based Computing Method, after t_1, t_2, ..., t_j have been added to S, when only considering the j tuples in S, we have

$$P(r(t_1) = i) = \sum_{u=1}^{n_1} \rho_{ui}^{(j)}(t_1) \tag{14}$$

for the discrete attribute-wise uncertain data model, and

$$P(r(t_1) = i) = \int_{-\infty}^{+\infty} \mathcal{F}_i^{(j)}(x)dx \tag{15}$$

for the continuous attribute-wise uncertain data model, and $\gamma^{(j)}(t_1)$ can be computed as

$$\gamma^{(j)}(t_1) = \sum_{i=1}^{j} \omega_i P(r(t_1) = i) \tag{16}$$

Then, we have

Proposition 1 $\gamma^{(1)}(t_1) \geq \gamma^{(2)}(t_1) \geq \ldots \geq \gamma^{(N)}(t_1)$.

Proof We prove the proposition under the discrete attribute-wise uncertain data model. For the continuous attribute-wise uncertain data model, the proof is put in Appendix B.

We need to prove $\gamma^{(j)}(t_1) \geq \gamma^{(j+1)}(t_1)$ for $1 \leq j \leq N - 1$.

In the Tuple Insertion Based Computing Method, after t_{j+1} is added to S, we have

$$\begin{aligned}
&\gamma^{(j+1)}(t_1)\\
&= \sum_{i=1}^{j+1} w_i P(r(t_1)=i)\\
&= \sum_{i=1}^{j+1} w_i \sum_{u=1}^{n_1} \rho_{ui}^{(j+1)}\\
&= \sum_{i=1}^{j+1} w_i \sum_{u=1}^{n_1} (P(s_{j+1} \le s_{1u})\rho_{ui}^{(j)} + P(s_{j+1} > s_{1u})\rho_{u(i-1)}^{(j)})\\
&= \sum_{i=1}^{j+1} w_i \sum_{u=1}^{n_1} ((1 - P(s_{j+1} > s_{1u}))\rho_{ui}^{(j)} + P(s_{j+1} > s_{1u})\rho_{u(i-1)}^{(j)})\\
&= \sum_{i=1}^{j+1} w_i \sum_{u=1}^{n_1} (\rho_{ui}^{(j)} + P(s_{j+1} > s_{1u})(\rho_{u(i-1)}^{(j)} - \rho_{ui}^{(j)}))\\
&= \sum_{i=1}^{j+1} w_i \sum_{u=1}^{n_1} \rho_{ui}^{(j)} + \sum_{i=1}^{j+1} w_i \sum_{u=1}^{n_1} P(s_{j+1} > s_{1u})(\rho_{u(i-1)}^{(j)} - \rho_{ui}^{(j)})\\
&= \sum_{i=1}^{j+1} w_i \sum_{u=1}^{n_1} \rho_{ui}^{(j)} + \sum_{u=1}^{n_1} P(s_{j+1} > s_{1u}) \sum_{i=1}^{j+1} w_i(\rho_{u(i-1)}^{(j)} - \rho_{ui}^{(j)})\\
&= \sum_{i=1}^{j} w_i \sum_{u=1}^{n_1} \rho_{ui}^{(j)} + \sum_{u=1}^{n_1} P(s_{j+1} > s_{1u}) \sum_{i=1}^{j} (w_{i+1} - w_i)\rho_{ui}^{(j)}
\end{aligned} \tag{17}$$

In the last step,

$$\sum_{i=1}^{j+1} w_i \sum_{u=1}^{n_1} \rho_{ui}^{(j)} = \sum_{i=1}^{j} w_i \sum_{u=1}^{n_1} \rho_{ui}^{(j)}$$

since $\rho_{u(j+1)}^{(j)} = 0$, and

$$\sum_{i=1}^{j+1} w_i(\rho_{u(i-1)}^{(j)} - \rho_{ui}^{(j)}) = \sum_{i=1}^{j} (w_{i+1} - w_i)\rho_{ui}^{(j)}$$

can be obtained by recombining the sum items and using the equalities $\rho_{u0}^{(j)} = 0$ and $\rho_{u(j+1)}^{(j)} = 0$.

Continuing, from

$$\sum_{i=1}^{j} w_i \sum_{u=1}^{n_1} \rho_{ui}^{(j)} = \gamma^{(j)}(t_1)$$

and

$$w_{i+1} \le w_i$$

we get

$$\gamma^{(j+1)}(t_1) \le \gamma^{(j)}(t_1) \tag{18}$$

□

Based on Proposition 1, for each new tuple t in T, we can compute $\gamma^{(1)}(t)$, $\gamma^{(2)}(t)$, ..., $\gamma^{(N)}(t)$ successively. If for some i $(1 \le i \le N-1)$, $\gamma^{(i)}(t)$ is less than the Kth largest PRF value retrieved so far, then, t must not be in the top-K list, and the computing of the exact PRF value for t is avoided; otherwise, $\gamma(t) = \gamma^{(N)}(t)$.

Example 4 We continue with the uncertain table in Fig. 1b. The goal is to find the top-2 tuples with the largest PRF values. Suppose that the weights are set as $\omega_1 = 1$, $\omega_2 = \frac{1}{2}$,

$\omega_3 = \frac{1}{3}$ and we have computed $\gamma(t_1) = 0.8$ and $\gamma(t_2) = 0.7$ by using the Tuple Insertion Based Computing Method. Now, consider t_3. We compute $\gamma^{(1)}(t_3)$, $\gamma^{(2)}(t_3)$ and $\gamma^{(3)}(t_3)$ successively. We start with a set S which only contains t_3, and then add t_1, t_2 in order. When S only contains t_3, we have $\gamma^{(1)}(t_3) = 1.0$. After t_2 is added to S, we have $\gamma^{(2)}(t_3) = 0.545$, computed according to the Tuple Insertion Based Algorithm. Since the second largest PRF value retrieved so far is 0.7 and $\gamma^{(2)}(t_3) < 0.7$, we can conclude that t_3 must not be in the top-2 list and the further computation is avoided.

Strategy 2 Recall that the support of a pmf or pdf f is $\{x | f(x) > 0\}$. For a tuple we call the support of its score's pmf or pdf its score support. For a tuple t, if there are n_L tuples whose score support lies to the left of of t's score support, and n_R tuples whose score support lies to the right of t's score support, then all the possible ranking places for t are $\{i | n_R < i \leq n - n_L\}$. We use T' to denote the tuples whose score support overlap with t's score support. Then, when computing the PRF value for t, only the tuples in T' need to be considered. We only need to set the weight corresponding to the ith place ($1 \leq i \leq n - n_L - n_R$) in T' as w_{n_R+i}, since the ith ranking place in T' corresponds to the $(n_R + i)$th ranking place in T, and use the same computing process as on T. The use of this pruning rule will reduce the time cost of computing $\gamma(t_i)$ from $O(M_{t_i}|T|^2)$ to $O(M_{t_i}|T'|^2)$ for the discrete attribute-wise uncertain data model, and from $O(M_{f_i}|T|^2)$ to $O(M_{f_i}|T'|^2)$ for the continuous attribute-wise uncertain data model. Here, M_{t_i} is the number of alternative scores of t_i, and M_{f_i} is the number of intervals divided by δ in the corresponding integrations by using Simpson method.

Strategy 3 We can sort all the tuples in T according to their left end point of score support, and denote the Kth largest left end point of score support as α. For each other tuple t, if its right end point of score support is less than α, then t must not be in the top-K list, and can be neglected. Thus, the time costs for computing the PRF values of all the tuples whose right end points of score supports are less than α are saved.

Based on the above three strategies, we develop our pruning as follows: We first find the Kth largest left end point α of the score supports of the tuples in T, and neglect all the tuples whose overall score supports fall to the left of α. All the remaining tuples are sorted according to their left end points of score supports, and stored in a list $\mathcal{L}$. Next, we construct a heap $\mathcal{H}$ from the first K tuples in $\mathcal{L}$ according to their PRF values. Then, we retrieve a new tuple t_{new} from $\mathcal{L}$ and compute the upper bound of its PRF value by using Strategy 1 and Strategy 2. If the upper bound is less than the lowest PRF value in $\mathcal{H}$, then t_{new} is not in the top-K list, and the computing of $\gamma(t_{new})$ is stopped. Otherwise, the computing process in Strategy 1 and Strategy 2 is continued until we can decide that t_{new} is not in the top-K list or the exact value of $\gamma(t_{new})$ is computed. If $\gamma(t_{new})$ is larger than the lowest PRF value in $\mathcal{H}$, the tuple with the lowest PRF value is deleted from $\mathcal{H}$ and t_{new} is inserted into $\mathcal{H}$. This process is repeated until all the tuples in $\mathcal{L}$ have been processed. Finally, all the tuples in $\mathcal{H}$ are sorted and returned to the user.

For the dataset with continuous value uncertainty, a small positive real number δ is used to divide the integral interval into subintervals when computing the corresponding integrals. Smaller δ will make the algorithm more accurate and more time consuming; larger δ will make the algorithm less accurate and less time consuming. It is difficult for a user to specify the value of δ. We take an adaptive strategy which can choose appropriate value for δ automatically. We first estimate each $\gamma(t_i)$ as $[lower_i, upper_i]$ by adopting an initial larger δ (the length of the smallest support of all the tuple pdfs) in the process of computing the integrals, and decide whether it is in the top-K list. For tuple t_i, we have three cases: (i) if $upper_i < lower_{Kth}$, where $lower_{Kth}$ is the lower bound for the Kth largest tuple retrieved so far, then tuple t_i must not be in the top-K list and can be neglected; (ii) if $lower_i > upper_{Kth}$, where $upper_{Kth}$ is the upper bound for the Kth largest tuple retrieved so far, then the original Kth largest tuple must not be in the top-K list and can be neglected, and tuple t_i is inserted into the top-K largest tuples retrieved so far; (iii) the estimated range for $\gamma(t_i)$ and the Kth largest tuple overlap, we cannot decide whether t_i should be in the top-K list. To insert t_i into the correct position in top-K for case (ii), or to distinguish t_i from the Kth largest tuple in case (iii), $\delta/2$ is used to divide the integral interval and the corresponding integrals are recomputed and more accurate estimates for the PRF values of the top-K tuples retrieved so far and $\gamma(t_i)$ are produced. This process is repeated until t_i is inserted into the correct position or can be distinguished from the Kth largest tuple.

Note that in the process of the estimate of $\gamma(t_i)$ as $[lower_i, upper_i]$, we adopt the adaptive Simpson method (Lyness 1969)

$$\epsilon \leq |(S(a,c) + S(c,b) - S(a,b))/15| \tag{19}$$

where $[a, b]$ is an interval with midpoint c, ϵ is the computing error for the integral on $[a, b]$, and $S(a, b)$, $S(a, c)$, $S(c, b)$ are the estimates given by Simpson's rule on the corresponding intervals. In (8), if $\int_{-\infty}^{+\infty} \mathcal{F}_1(x)dx$, $\int_{-\infty}^{+\infty} \mathcal{F}_2(x)dx, \ldots,$ $\int_{-\infty}^{+\infty} \mathcal{F}_u(x)dx$ have been estimated as $[A_1, B_1]$, $[A_2, B_2]$, $\ldots$, $[A_u, B_u]$ respectively, then $\gamma^{(u)}(t_i)$ can be estimated as $[\sum_{j=1}^{u} \omega_j A_j, \sum_{j=1}^{u} \omega_j B_j]$. $\int_{-\infty}^{+\infty} \mathcal{F}_1(x)dx$, $\int_{-\infty}^{+\infty} \mathcal{F}_2(x)dx, \ldots,$ $\int_{-\infty}^{+\infty} \mathcal{F}_u(x)dx$ can be estimated recursively according to (9) and (10).

5 Dealing with top-K aggregate queries

In this section, we formulate the top-K aggregate query on dataset with value uncertainty based on the PRFs ranking framework, and then apply the pruning algorithm described in Section 4 to computing the query.

The top-K aggregate query on dataset with value uncertainty is formulated as follows:

Suppose that T is a table with tuples $t_1, t_2, \ldots, t_n$, and A is a set of grouping attributes. In this paper, we assume that the grouping attributes in A are all deterministic. Each tuple t_i has an uncertain score s_i. For the discrete attribute-wise uncertain data model, s_i is described as a **pmf**; for the continuous attribute-wise uncertain data model, s_i is described as a **pdf**. Given an aggregate function (**sum**, **avg**, **count**, **max**, or **min**), a top-k aggregate query on T first divides all its tuples into m groups g_1, g_2, $\ldots$, g_m according to A, computes the aggregates for each group, and then returns the K groups with the largest aggregates.

Figure 7a illustrates an example table $Movie$ with 10 tuples, each of which describes a movie with a year attribute and an uncertain score attribute representing the movie's rating. Figure 7b illustrates an example top-K aggregate query issued on table $Movie$: *"find the top 2 years with the highest average movie ratings"*.

Since the scores of the tuples in each group are random variables, the aggregates are also random variables. Under the discrete attribute-wise uncertain data model, for each group g containing $|g|$ tuples t_1, t_2, $\ldots$, $t_{|g|}$ with scores s_1, s_2, $\ldots$, $s_{|g|}$ described as **pmf**s f_1, f_2, $\ldots$, $f_{|g|}$, the aggregate values for **sum**, **avg**, **max**, and **min** can be derived as **pmf**s from f_1, f_2, $\ldots$, $f_{|g|}$ according to the classic probability theory (DeGroot and Schervish 2011), described as follows: **sum**: The **pmf** of the **sum** of s_1 and s_2 can be computed as:

$$f_{s_1+s_2}(x) = \sum_{z=-\infty}^{+\infty} f_1(z) f_2(x-z) \tag{20}$$

Name	Year	Score
Movie1	1981	$\{\langle 1,0.1\rangle, \langle 2,0.1\rangle, \langle 3,0.2\rangle, \langle 4,0.2\rangle, \langle 5,0.4\rangle\}$
Movie2	1981	$\{\langle 4,0.6\rangle, \langle 5,0.4\rangle\}$
Movie3	1982	$\{\langle 1,0.1\rangle, \langle 2,0.9\rangle\}$
Movie4	1982	$\{\langle 1,0.2\rangle, \langle 2,0.2\rangle, \langle 3,0.3\rangle, \langle 4,0.3\rangle\}$
Movie5	1982	$\{\langle 1,0.6\rangle, \langle 2,0.1\rangle, \langle 3,0.1\rangle, \langle 4,0.1\rangle, \langle 5,0.1\rangle\}$
Movie6	1983	$\{\langle 1,0.8\rangle, \langle 2,0.2\rangle\}$
Movie7	1983	$\{\langle 4,0.7\rangle, \langle 5,0.3\rangle\}$
Movie8	1983	$\{\langle 1,0.1\rangle, \langle 2,0.1\rangle, \langle 3,0.2\rangle, \langle 4,0.1\rangle, \langle 5,0.5\rangle\}$
Movie9	1984	$\{\langle 1,0.8\rangle, \langle 2,0.1\rangle, \langle 3,0.1\rangle\}$
Movie10	1984	$\{\langle 3,0.2\rangle, \langle 4,0.2\rangle, \langle 5,0.6\rangle\}$

(a) Table Movie

```
SELECT Year, avg(Score)
FROM Movie
GROUP BY Year
ORDER BY avg(Score)
DESC
LIMIT 2
```

(b) Top-K aggregate query

Fig. 7 An example top-K aggregate query issued on a table with score attributes described as **pmf**s

The **pmf** of the **sum** of s_1, s_2, …, $s_{|g|}$, denoted as $f_{s_1+s_2+\ldots+s_{|g|}}$, can be computed inductively.

avg: The **pmf** of the **avg** of $s_1, s_2, \ldots, s_{|g|}$ can be computed as:

$$f_{\frac{s_1+s_2+\ldots+s_{|g|}}{|g|}}(x) = f_{s_1+s_2+\ldots+s_{|g|}}(|g|x) \tag{21}$$

max:The **cmf** (cumulative distribution function) of the **max** of s_1, s_2, …, $s_{|g|}$ can be computed as:

$$P(max(s_1, s_2, \ldots, s_{|g|}) \leq x) = P(s_1 \leq x)P(s_2 \leq x) \cdots P(s_{|g|} \leq x) \tag{22}$$

min:The **cmf** of the **min** of $s_1, s_2, \ldots, s_{|g|}$ can be computed as:

$$\begin{aligned} P(min(s_1, s_2, \ldots, s_{|g|}) \leq x) = &\ 1 - (1 - P(s_1 \leq x))(1 - P(s_2 \leq x)) \\ &\cdots (1 - P(s_{|g|} \leq x)) \end{aligned} \tag{23}$$

The **pmf** can be computed from the corresponding **cmf** by

$$P(X = x) = P(X \leq x) - P(X < x) \tag{24}$$

where X is random variable representing the **max** or **min** of $s_1, s_2, \ldots, s_{|g|}$.

Under the continuous attribute-wise uncertain data model, for each group g containing $|g|$ tuples $t_1, t_2, \ldots, t_{|g|}$ with scores $s_1, s_2, \ldots, s_{|g|}$ described as probability density functions $f_1, f_2, \ldots, f_{|g|}$, the aggregate values for **sum**, **avg**, **max**, **min** are derived as probability density functions (called group aggregate pdf or aggregate pdf) from $f_1, f_2, \ldots, f_{|g|}$, described as follows:

sum: The pdf of the **sum** of s_1 and s_2 can be computed as:

$$f_{s_1+s_2}(x) = \int_{-\infty}^{+\infty} f_1(y) f_2(x-y) dy \tag{25}$$

The pdf of the **sum** of $s_1, s_2, \ldots, s_{|g|}$, denoted as $f_{s_1+s_2+\ldots+s_{|g|}}$ can be computed inductively.

avg: The pdf of the **avg** of $s_1, s_2, \ldots, s_{|g|}$ can be computed from $f_{s_1+s_2+\ldots+s_{|g|}}$ as follows:

$$f_{\frac{s_1+s_2+\ldots+s_{|g|}}{|g|}}(x) = |g| f_{s_1+s_2+\ldots+s_{|g|}}(|g|x) \tag{26}$$

max: The pdf of the **max** of s_1, $s_2, \ldots, s_{|g|}$ can be computed as:

$$f_{\max(s_1, s_2, \ldots, s_{|g|})} = \sum_{1 \leq i \leq |g|} (f_i(x) \prod_{1 \leq j \leq |g|, j \neq i} \int_{-\infty}^{x} f_j(t) dt) \tag{27}$$

min: The pdf of the **min** of $s_1, s_2, \ldots, s_{|g|}$ can be computed as:

$$f_{\min(s_1, s_2, \ldots, s_{|g|})} = \sum_{1 \leq i \leq |g|} (f_i(x) \prod_{1 \leq j \leq |g|, j \neq i} (1 - \int_{-\infty}^{x} f_j(t) dt)) \tag{28}$$

The **count** aggregate has the same semantic as in the traditional database. Hence, in the following, we mainly focus on the above four aggregates.

Figure 8a shows an example table T with uncertain attribute S, and Fig. 8b shows an example top-K aggregate query issued on T. After dividing the tuples in T into groups g_1, g_2 and g_3 according to attribute R, and computing the **pmf** for each group, an intermediate result table is produced as shown in Fig. 8c. Each group can be considered as a tuple with a random score, and the corresponding algorithms for selecting the top-K tuples can be applied with tuples replaced as groups. The Tuple Insertion Based Computing Method for the top-K ranking query becomes Group Insertion Based Computing Method for the top-K aggregate query and the corresponding pruning algorithm can be applied directly as in Section 4. Note that the left end point and right end point of the aggregate **pmf** or **pdf** support for each group can be computed as in Table 1, without computing the corresponding aggregates. Here, for each group g, $f_j(1 \leq j \leq |g|)$ denotes the **pmf**s or **pdf**s describing the scores of its $|g|$ tuples, and $L_{supp(f_j)}$ and $R_{supp(f_j)}$ denote the left end point and right end point of the support of f_i respectively.

Recall that, under the discrete attribute-wise uncertain data model, the time cost for computing the PRF value for a group is $O(M^3N^2)$ for the Model Transforming Based Algorithm, and $O(MN^2)$ for the Group Insertion Based Algorithm, where N is the number of groups and M is the average number of alternatives for each group's score. The **sum** and **avg** aggregates usually produce large amount of possible scores for each group. For example, a group g containing $|g|$ tuples $t_1, t_2, \ldots, t_{|g|}$ with $M_1, M_2, \ldots, M_{|g|}$ possible scores respectively will produce $M_1 \times M_2 \times \ldots \times M_n$ possible alternative scores for the **sum** and **avg** aggregates in the worst case. This makes the Group Insertion Based Algorithm and corresponding pruning algorithm more preferable compared with the Model Transforming Based Algorithm. For the **sum** and **avg** aggregates, for a group, when $M_1 \times M_2 \times \ldots \times M_n > 100$, we adopt the Monte-Carlo approximate algorithm. Under the continuous attribute-wise uncertain data model, suppose that there are N groups, and for each group g with aggregate **pdf** f_g, M_{f_g} is the number of subintervals of the support of f_g divided by δ in the Simpson method, the time cost of computing the PRF value of group g by using the Group Insertion Based Algorithm is $O(M_{f_g}N^2)$, which is preferable to the cubic *generating functions* based method.

	R	S
t_1	a	$\{\langle 1, 0.2\rangle, \langle 2, 0.8\rangle\}$
t_2	a	$\{\langle 3, 0.1\rangle, \langle 4, 0.9\rangle\}$
t_3	b	$\{\langle 2, 0.3\rangle, \langle 3, 0.7\rangle\}$
t_4	b	$\{\langle 4, 0.4\rangle, \langle 5, 0.6\rangle\}$
t_5	c	$\{\langle 2, 0.2\rangle, \langle 3, 0.8\rangle\}$

(a)

```
SELECT R, sum(S)
FROM T
GROUP BY R
ORDER BY sum(S)
LIMIT 2
```

(b)

	R	Aggregate
g_1	a	$\{\langle 4, 0.02\rangle, \langle 5, 0.26\rangle, \langle 6, 0.72\rangle\}$
g_2	b	$\{\langle 6, 0.12\rangle, \langle 7, 0.46\rangle, \langle 8, 0.42\rangle\}$
g_3	c	$\{\langle 2, 0.2\rangle, \langle 3, 0.8\rangle\}$

(c)

Fig. 8 **a** An example table T with uncertain attribute S **b** An example top-K aggregate query issued on T **c** the intermediate result table after dividing the tuples in T into groups g_1, g_2, and g_3, and computing the corresponding **sum** aggregate

Table 1 Left end point and right end point of the aggregate **pmf** or **pdf** support for group g

Aggregate	Left end point	Right end point
sum	$\sum_{j=1}^{\lvert g\rvert} L_{supp(f_j)}$	$\sum_{j=1}^{\lvert g\rvert} R_{supp(f_j)}$
avg	$\frac{\sum_{j=1}^{\lvert g\rvert} L_{supp(f_j)}}{\lvert g\rvert}$	$\frac{\sum_{j=1}^{\lvert g\rvert} R_{supp(f_j)}}{\lvert g\rvert}$
max	$\max_{j=1}^{\lvert g\rvert} L_{supp(f_j)}$	$\max_{j=1}^{\lvert g\rvert} R_{supp(f_j)}$
min	$\min_{j=1}^{\lvert g\rvert} L_{supp(f_j)}$	$\min_{j=1}^{\lvert g\rvert} R_{supp(f_j)}$

6 Experiments

To evaluate the performance of our pruning algorithm, under the discrete attribute-wise uncertain data model, for the top-K ranking query, we implemented the Model Transforming Based Algorithm, the Tuple Insertion Based Algorithm, and the Upper Bound Estimation Based Pruning Algorithm; for the top-K aggregate query, we implemented the **sum**, **avg**, **max** and **min** aggregates, and applied the Model Transforming Based Algorithm, the Group Insertion Based Algorithm, and the Upper Bound Estimation Based Pruning Algorithm respectively. Under the continuous attribute-wise uncertain data model, we implemented the Generating Function Based Algorithm, corresponding to the Model Transforming Based Algorithm in the discrete case. The other corresponding algorithms with the same names are also implemented. Our algorithms were implemented in GNU C++. We conducted a systematic empirical study using both real datasets and synthetic datasets. All the experiments were run on a PC with double 2.20 GHz CPUs and 2GB RAM, running Linux operating system.

We have verified that all the algorithms produce the same top-K answer list on both the real datasets and the synthetic datasets. This observation empirically verifies the correctness of the Tuple (Group) Insertion Based Algorithm and the Upper Bound Estimation Based Pruning Algorithm. In the following, we focus on the performance study.

For the parameterized ranking function in (1), we experimented with three forms of weights: (1) randomly generated weights, reciprocal weights, and probabilistic threshold top-K weights. For the randomly generated weights, we generated N random numbers and sorted them into a descending order and ω_i is set to the ith number; for the reciprocal weights, we set $\omega_i = \frac{1}{i}$; for the probabilistic threshold top-K weights, we set $\omega_1 = 1, \ldots, \omega_K = 1, \omega_{K+1} = 0, \ldots, \omega_N = 0$. In our experiments, we found that the performance of our pruning algorithm for the randomly generated weights is similar to that using the reciprocal weights, and is worse than the probabilistic top-K weights. In the following, we only report the results for the randomly generated weights.

6.1 Results under the discrete attribute-wise uncertain data model

We first present the experimental results under the discrete attribute-wise uncertain data model. In the following, all the time axes adopt the logarithmic coordinates.

6.1.1 Results on real datasets

To investigate the performance of our pruning algorithm in real applications, we use the MovieLens Data[1], which contains the following three datasets with three different scales:

[1] http://www.grouplens.org

(1) MovieLens 100k (consisting of 100,000 ratings from 943 users on 1682 movies), (2) MovieLens 1M (consisting of 1,000,209 ratings from 6,040 users on 3,706 movies) and (3) MovieLens 10M (consisting of 10,000,054 ratings from 71,567 users on 10,677 movies). All the ratings are made on a 5-star scale. The ratings of the first two data sets have whole-star increments, producing 5 different scales. The ratings of the third dataset have half-star increments, producing 10 different scales. Based on the three datasets, we produced three uncertain tables: Movie1, Movie2 and Movie3 with 1682 tuples, 3706 tuples, 10677 tuples respectively. Each tuple represents a movie and has an uncertain score attribute representing the movie's rating. Each score attribute has 5 possible ratings and corresponding probabilities for tables Movie1 and Movie2, and 10 possible ratings and corresponding probabilities for table Movie3. The probability that a movie takes a rating is computed as the proportion of the users who give the corresponding rating to the movie.

Figure 9 compares the running times of the Model Transforming Based Algorithm, the Tuple Insertion Based Algorithm and the Upper Bound Estimation Based Pruning Algorithm for the top-K ranking query on the three uncertain tables Movie1, Movie2 and Movie3. We set $K = 10$. The running time of the Tuple Insertion Based Algorithm is always less than the Model Transforming Based Algorithm, and the Upper Bound Estimation Based Pruning Algorithm outperforms the Tuple Insertion Based Algorithm by 2, 3 and 5 orders of magnitude for the three uncertain tables respectively.

Figure 10 shows the running times of the Model Transforming Based Algorithm, the Tuple Insertion Based Algorithm and the Upper Bound Estimation Based Pruning Algorithm for the top-K ranking query on table Movie3 with respect to different number of K. The time costs of all the algorithms are linear to the number of K.

The corresponding performance studies of the pruning algorithm for the top-K aggregate query are showed in Fig. 11. The top-K aggregate query is *"Find the top-K years with the largest average ratings"*. We also substitute the other three aggregates **sum**, **max** and **min** for the **avg** aggregate. Since the performances for the **avg** and **min** aggregates are similar to the **sum** and **max** aggregates respectively, we only present the results for the **sum** and **max** aggregates. In Fig. 11, K is set to 10. We can see that, for the top-K aggregate query, the

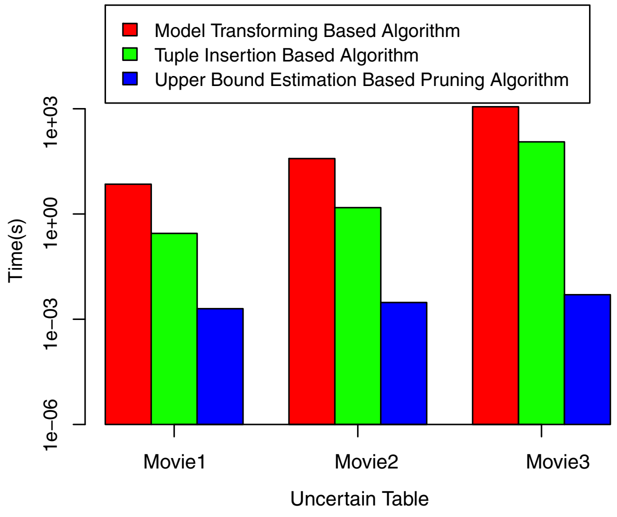


Fig. 9 Running times of the algorithms for the top-K ranking query on uncertain tables Movie1, Movie2, and Movie3

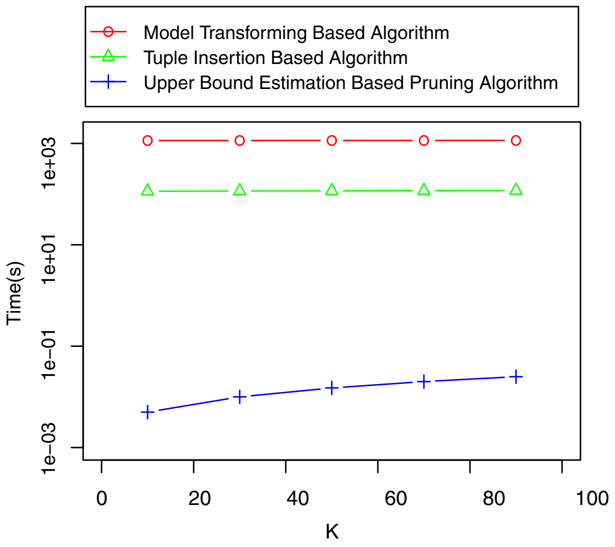


Fig. 10 Running times of the algorithms for the top-K ranking query on Movie3 with respect to different number of K

running time of the Group Insertion Based Algorithm is also always less than the Model Transforming Based Algorithm, and the Upper Bound Estimation Based Pruning Algorithm outperforms the Group Insertion Based Algorithm by up to 4 orders of magnitude. The time costs of all the algorithms are also linear to the number of K, and we omit the corresponding figures.

6.1.2 *Results on synthetic datasets*

In order to evaluate the impact of datasets with different characteristics on both the score value distribution and the probability distribution, we generated various synthetic data sets. Note that, in this paper, we mean by "score value distribution" the set of candidate values that appear in an uncertain value, and "probability distribution" the corresponding probability degrees. For the tuple scores, the uniform distribution and the normal distribution are

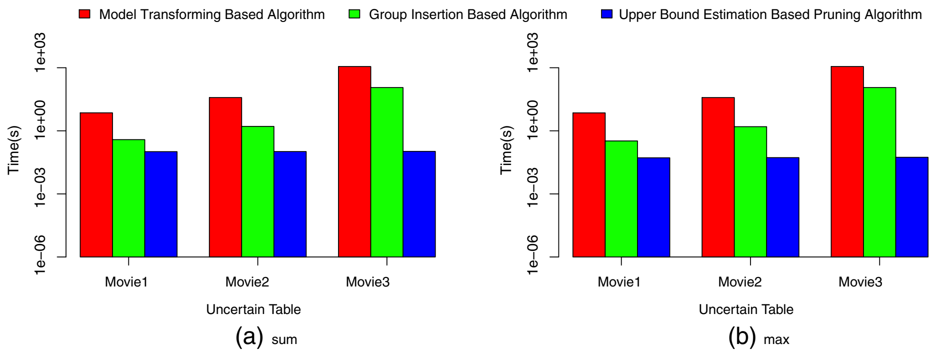


Fig. 11 Running times of the algorithms for the top-K aggregate query on uncertain tables Movie1, Movie2 and Movie3

used. For the score probabilities, the uniform distribution and the gamma distribution are used. Combining the score distribution and probability distribution, four data sets are generated: uniform-uniform, uniform-gamma, normal-uniform, and normal-gamma, meaning the concatenation of the score distribution and then the probability distribution. To produce a skewed distribution for the probabilities, the shape parameter and rate parameter of the gamma distribution are set to 0.1 and 1 respectively.

Figure 12 shows the running times of the algorithms with respect to different number of tuples on the four synthetic datasets with different score distributions and probability distributions. Each tuple has a fixed number of 10 alternatives. Again, the running time of the Tuple Insertion Based Algorithm is always less than the Model Transforming Based Algorithm, and the Upper Bound Estimation Based Pruning Algorithm outperforms the Tuple Insertion Based Algorithm up to 5 orders of magnitude. The Upper Bound Estimation Based Pruning Algorithm is most effective for the uniform-gamma dataset, this is because the uniform distribution produces scores with a lower rate of overlap than the normal distribution and the gamma distribution is more skewed than the uniform distribution, and thus more tuples can be guaranteed not to be in top-K without computing their exact PRF values, as verified in Fig. 13.

Figure 14 shows the running times of the algorithms for the top-K ranking query on the uniform-uniform synthetic dataset with respect to different number of possible alternatives. This verifies that, under different number of possible alternatives in each tuple, the Tuple Insertion Based Algorithm outperforms the Model Transforming Based Algorithm, and the Upper Bound Estimation Based Pruning Algorithm further improve the performance of the top-K ranking query. We got the same results for the other three datasets.

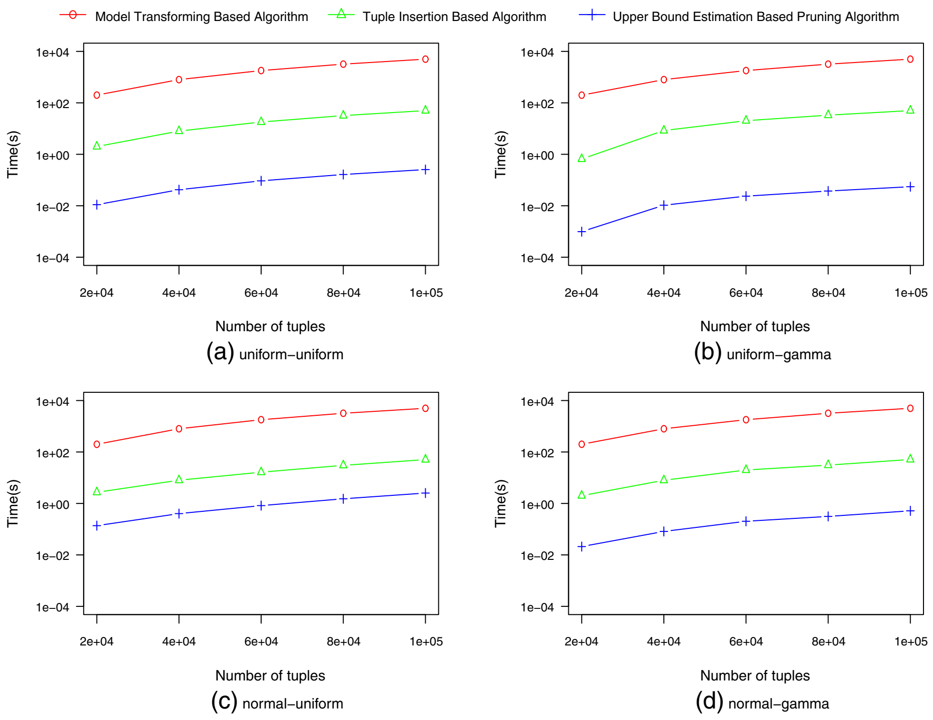


Fig. 12 Running times of the algorithms for the top-K ranking query with respect to different number of tuples on synthetic data under the discrete attribute-wise uncertain data model

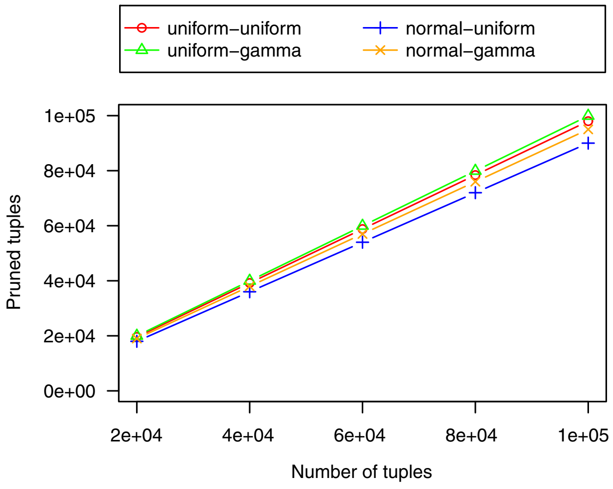


Fig. 13 Number of tuples that can be decided not to be in top-K for the Upper Bound Estimation Based Pruning Algorithm on synthetic data under the discrete attribute-wise uncertain data model

Figure 15 shows the running times of the algorithms for the top-K aggregate query with respect to different number of tuples in each group on the uniform-uniform synthetic dataset. The data set has a fixed number of 10000 groups and each tuple has 10 possible alternatives. We set $K = 10$. For the **sum** aggregate, since the number of alternatives of the **pmf** of each group is exponential to the number of tuples in it, we use the Monte-Carlo Approximate algorithm and select the 100 alternatives with the largest probabilities. All other alternatives' probabilities are added to the nearest alternatives. This produces up to 100 alternatives for each group. For the **sum** aggregate in Fig. 15, different tuples in a group all produces 100 alternatives for each group's **pmf**, so the time costs remain nearly constant. However, the

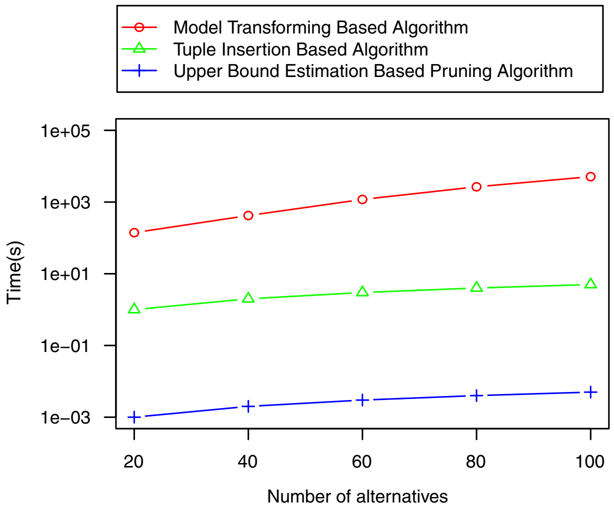


Fig. 14 Running times of the algorithms for the top-K ranking query with respect to different number of possible alternatives on synthetic data under the discrete attribute-wise uncertain data model

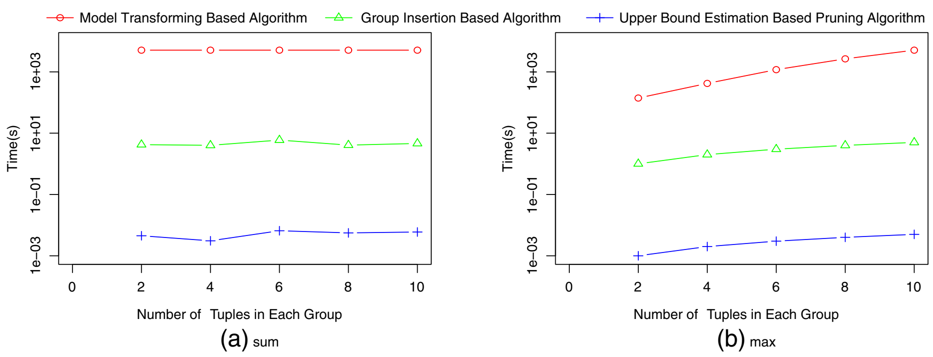


Fig. 15 Running times of the algorithms for the top-K aggregate query with respect to different number of tuples in each group on the uniform-uniform synthetic dataset

Upper Bound Estimation Based algorithm takes only 5ms, while the Model Transforming based algorithm takes about 1.5 hours. For the **max** aggregate, after the number of tuples in each group increases to 10, we got the same running times as the **sum** aggregate. The **avg** and **min** aggregates are similar to the **sum** and **max** aggregates respectively.

We also investigated the running times of the algorithms for the top-K aggregate queries on synthetic data sets with respect to different number of groups, different number of alternatives for each tuple's score, and different number of K, and arrived the similar conclusion that the Group Insertion Based Algorithm outperforms the Model Transforming Based Algorithm, and the Upper Bound Estimation Based Pruning Algorithm outperforms the Group Insertion Based Algorithm by up to 5 orders of magnitude.

6.2 Results under the continuous attribute-wise uncertain data model

Next, we present the experimental results under the continuous attribute-wise uncertain data model. We mainly use several synthetic datasets with various continuous distributions to study our algorithms. In each experiment, a small number $delta$ is used to divide the integral interval into subintervals. We choose $delta$ in the adaptive approach as described in Section 4.

We synthesize a dataset containing 1000000 tuples. For each tuple, its score **pdf** is randomly selected from the four commonly used distributions: uniform distribution, normal distribution, gamma distribution and beta distribution. When a distribution is selected, its parameters (for example, the mean and variance of the normal distribution) are all generated uniformly from [0, 1000]. The supports of the uniform distribution and the beta distribution are bounded naturally. The supports of all the normal distributions are truncated to $S = [\mu - r, \mu + r]$ such that $P(S) = 0.9999$, and the supports of all the gamma distributions are truncated to $S = [0, r]$ such that $P(S) = 0.9999$.

Figure 16 shows the running times of the algorithms for top-K ranking query with respect to different number of tuples on synthetic data under the continuous attribute-wise uncertain data model. This verifies that, under the continuous attribute-wise uncertain data model, the time cost of the Tuple Insertion Based algorithm is lower than the Generating Function Based Algorithm, and after adopting the pruning method, the time cost is further reduced largely. Figure 17 shows the the running times of the algorithms for top-K ranking query with respect to different number of K on synthetic data under the continuous attribute-wise uncertain data model, verifying that all the algorithms are linear to K.

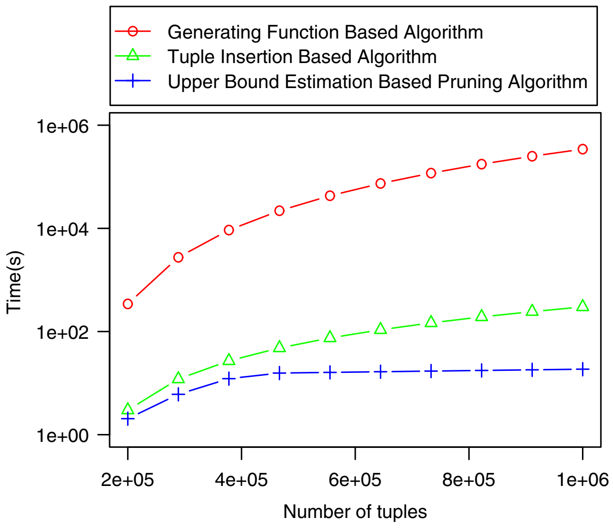


Fig. 16 Running times of the algorithms for the top-K ranking query with respect to different number of tuples on synthetic data under the continuous attribute-wise uncertain data model

We also compare the time costs of the algorithms for the top-K aggregate query under the continuous attribute-wise uncertain data model. Figure 18 shows the running times with respect to different number of groups for the **sum** and **max** aggregate respectively. All the 1000000 tuples are divided into 10000 groups, with 100 tuples in each group. We can see

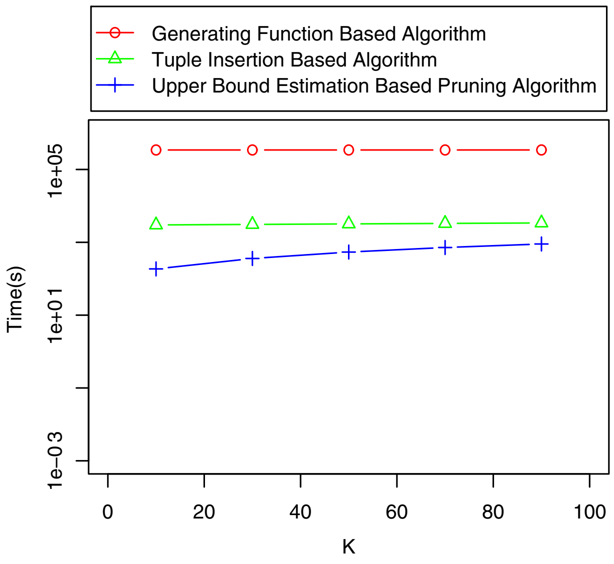


Fig. 17 Running times of the algorithms for the top-K ranking query with respect to different number of K on synthetic data under the continuous attribute-wise uncertain data model

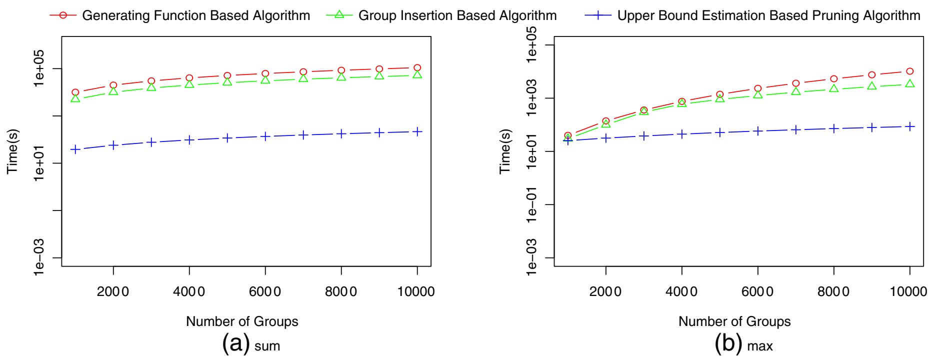


Fig. 18 Running times of the algorithms for the top-K aggregate query with respect to different number of groups on synthetic data under the continuous attribute-wise uncertain data model

that the time cost of the Group Insertion Based Algorithm is lower than the Generating Function Based Algorithm, and the Upper Bound Estimation Based Pruning Algorithm further reduced the time cost largely, with an up to 3 orders of magnitude improvement.

7 Related work

Top-K ranking queries report the top-K tuples according to a ranking function defined on one or more score attributes. A survey on top-K ranking in traditional relational database can be found in (Ilyas et al. 2008).

Recently, ranking uncertain data has drawn many research interests, with different ranking semantics proposed, such as tuple probability based rank (Ré et al. 2007), **U-Top** and **U-Rank** (Soliman et al. 2007), probabilistic threshold top-K (Hua et al. 2008), **Global-Top$k$** (Zhang and Chomicki 2009), **expected rank** (Jestes et al. 2011), and **typical top-K** queries (Ge et al. 2009). Yi et al.'s work aims to improve the ranking performance under the **U-Top** and **U-Rank** semantics. Recently, Li et al. (Li et al. 2009) present a unified ranking approach based on the PRF ranking functions, which can generalize many previously proposed semantics. Wang et al. (Wang et al. 2011) further present efficient pruning techniques for Top-K ranking in databases with the tuple-wise uncertain data model under the PRFs based ranking framework. However, this cannot apply to efficient top-K ranking under the attribute-wise uncertain data model, which is the aim of our work.

The aforementioned research work mainly focused on discrete uncertainty. Soliman and Ilyas (Soliman and Ilyas 2009) considered the problem of ranking continuous uncertain data, considering **UTop-Rank**, **UTop-Prefix**, **UTop-Set**, and **Rank-Agg** semantics. Probabilistic ranked (**PRank**) query and probabilistic inverse ranking (**PIR**) query are formulated and tackled in (Lian and Chen 2008) and (Lian and Chen 2011) respectively. Jian Li and Amol Deshpande (Li and Deshpande 2010) further presented a **PRF** based framework for ranking continuous uncertain data, which can simulate or approximate a variety of ranking functions with appropriate parameter choices. However, their algorithms are cubic. In this paper, we present a quadratic algorithm for ranking continuous uncertain data under the PRFs based framework, based on which efficient pruning strategies are further developed to reduce the time cost. Cheng et al. (Cheng et al. 2003) studied aggregate queries on continuous uncertain data. However, top-K aggregate queries were not considered in their work.

Soliman and Ilyas (Soliman and Ilyab 2008) consider the top-K aggregate queries on uncertain dataset with tuple-wise model under the **U-Top** and **U-Rank** semantics. Parallelizing with their work, this paper considered the top-K aggregate queries on uncertain dataset with attribute-wise model under the most recent PRFs based ranking framework. A complex query type is investigated in (Song et al. 2013) where the aggregate queries can be issued on a top-K result. This represents a different semantic from the top-K aggregate query investigated in this paper.

8 Conclusion

In this paper, we present a pruning algorithm for top-K ranking on dataset with value uncertainty. Experiments on both real data and synthetic data demonstrate the efficiency of the algorithm. We have made a simple assumption that each tuple has only one score attribute represented as a **pmf** or **pdf**. This can be easily extended to the general case where a ranking function of multiple score attributes is used since the **pmf** or **pdf** of the ranking function can be derived from the **pmf**s or **pdf**s of the corresponding score attributes and then the pruning algorithm can be applied. Our future work will be applying our algorithm to real world applications.

Appendix A: Tuple Insertion Based Computing Method and Possible World Semantics

Proposition A1 *The Tuple Insertion Based Computing Method conforms to the possible world semantics.*

Proof We only need to prove that for each tuple t, the Tuple Insertion Based Computing Method produces the same value for $P(r(t) = i)$ as the possible world semantics. In the following, we use t_1 as an example. Suppose that the score of t_1 is $\{\langle s_{11}, p_{11}\rangle, \langle s_{12}, p_{12}\rangle, \ldots, \langle s_{1n_1}, p_{1n_1}\rangle\}$.

We use Mathematical Induction on the number of tuples N.

First consider the case N = 1. According to the Tuple Insertion Based Computing Method,

$$P(r(t_1) = 1) = \sum_{u=1}^{n_1} \rho_{u1}^{(1)} = \sum_{u=1}^{n_1} p_{1u} = 1.0 \tag{A1}$$

According to the possible world semantics, there are n_1 possible worlds, and t_1 is ranked at the first place in all the possible worlds, we have

$$P(r(t_1) = 1) = \sum_{pw \in PW \wedge r_{pw}(t)=1} P(pw) = 1.0 \tag{A2}$$

The Tuple Insertion Based Computing Method produces the same computing result as the possible world semantics.

Next, suppose that when $N = j$, the proposition is true for the tuple set $S = \{t_1, t_2, \ldots, t_j\}$. When only considering the j tuples in S, we denote the corresponding set of possible worlds as $PW^{(j)}$. We have

$$\rho_{u(v-1)}^{(j)} = \sum_{pw \in PW^{((j)} \wedge V_{pw}(t_1)=s_{1u} \wedge r_{pw}(t_1)=v-1} P(pw) \tag{A3}$$

and

$$\rho_{uv}^{(j)} = \sum_{pw \in PW^{(j)} \wedge V_{pw}(t_1)=s_{1u} \wedge r_{pw}(t_1)=v} P(pw) \tag{A4}$$

where $V_{pw}(t_1)$ denote the score of t_1 in the possible world pw. After t_{j+1} is added to S, we have

$$\rho_{uv}^{(j+1)} = \sum_{pw \in PW^{(j+1)} \wedge V_{pw}(t_1)=s_{1u} \wedge r_{pw}(t_1)=v} P(pw) \tag{A5}$$

We verify that

$$\rho_{uv}^{(j+1)} = P(s_{j+1} \le s_{1u})\rho_{uv}^{(j)} + P(s_{j+1} > s_{1u})\rho_{u(v-1)}^{(j)} \tag{A6}$$

For each possible world $pw \in PW^{(j+1)}$, if the score of t_1 in pw is s_{1u} and t_1 is ranked at the vth position, then pw must be obtained from a possible world pw' in $PW^{(j)}$ by adding t_{j+1} with the following two cases: (1) $s_{j+1} \le s_{1u}$ and t_1 is ranked at the vth position in pw'; (2) $s_{j+1} > s_{1u}$ and t_1 is ranked at the $(v-1)$th position in pw'.

Continuing with (A5), we have

$$\begin{aligned}
&\rho_{uv}^{(j+1)} \\
&= \sum_{pw \in PW^{(j+1)} \wedge V_{pw}(t_1)=s_{1u} \wedge r_{pw}(t_1)=v \wedge s_{j+1} \le s_{1u}} P(pw) \\
&+ \sum_{pw \in PW^{(j+1)} \wedge V_{pw}(t_1)=s_{1u} \wedge r_{pw}(t_1)=v \wedge s_{j+1} > s_{1u}} P(pw) \\
&= \sum_{pw' \in PW^{(j)} \wedge V_{pw'}(t_1)=s_{1u} \wedge r_{pw'}(t_1)=v \wedge s_{(j+1)q} \le s_{1u}} P(pw')p_{(j+1)q} \\
&+ \sum_{pw' \in PW^{(j)} \wedge V_{pw'}(t_1)=s_{1u} \wedge r_{pw'}(t_1)=v-1 \wedge s_{(j+1)q} > s_{1u}} P(pw')p_{(j+1)q} \\
&= (\sum_{pw' \in PW^{(j)} \wedge V_{pw'}(t_1)=s_{1u} \wedge r_{pw'}(t_1)=v} P(pw'))(\sum_{s_{(j+1)q} \le s_{1u}} p_{(j+1)q}) \\
&+ (\sum_{pw' \in PW^{(j)} \wedge V_{pw'}(t_1)=s_{1u} \wedge r_{pw'}(t_1)=v-1} P(pw'))(\sum_{s_{(j+1)q} > s_{1u}} p_{(j+1)q}) \\
&= P(s_{j+1} \le s_{1u})\rho_{uv}^{(j)} + P(s_{j+1} > s_{1u})\rho_{u(v-1)}^{(j)}
\end{aligned} \tag{A7}$$

Hence, (A6) conforms to the possible world semantics.

Finally, from (A4), we can verify that

$$P(r(t) = i) = \sum_{u=1}^{n_1} \rho_{ui}^{(j+1)} \tag{A8}$$

conforms to the possible world semantics. □

Appendix B: The pruning Strategy 1 is also valid for the continuous uncertain data model

Proposition B1 *Proposition 1 is also true under the continuous attribute-wise uncertain data model.*

Proof We need to prove $\gamma^{(j)}(t_1) \geq \gamma^{(j+1)}(t_1)$ for $1 \leq j \leq N-1$.

Suppose that when $t_1, t_2, \ldots, t_u$ are considered, we have computed $\gamma^{(u)}(t_1)$ as

$$\gamma^{(u)}(t_1) = \sum_{i=1}^{u} \omega_i P(r(t_1)=i) = \sum_{i=1}^{u} \omega_i \int_{-\infty}^{+\infty} \mathcal{F}_i^{(u)}(x)dx \tag{B1}$$

When t_{u+1} is added, $\gamma^{(u+1)}(t_1)$ can be computed as

$$\gamma^{(u+1)}(t_1) \\ = \sum_{i=1}^{u}(\omega_i \int_{-\infty}^{+\infty} \mathcal{F}_i^{(u)}(x) \int_{-\infty}^{x} f_{u+1}(t)dtdx + \omega_{i+1} \int_{-\infty}^{+\infty} \mathcal{F}_i^{(u)}(x)(1-\int_{-\infty}^{x} f_{u+1}(t)dt)dx) \tag{B2}$$

From (B2), we can further get

$$\gamma^{(u+1)}(t_1) = \gamma^{(u)}(t_1) + \sum_{i=1}^{u}(\omega_{i+1}-\omega_i) \int_{-\infty}^{+\infty} \mathcal{F}_i^{(u)}(x)(1-\int_{-\infty}^{x} f_{u+1}(t)dt)dx \leq \gamma^{(u)}(t_1) \tag{B3}$$

□

References

Aggarwal, C.C., & Yu, P.S. (2009). *A survey of uncertain data algorithms and applications*: TKDE.

Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S., Sugihara, T., & Widom, J. (2006). Trio: A system for data, uncertainty, and lineage. In *VLDB*.

Cheng, R., Kalahnikov, D.V., & Prabhakar, S. (2003). Evaluating probabilistic queries over imprecise data. In *SIGMOD*.

DeGroot, M.H., & Schervish, M.J. (2011). *Probability and Statistics*, 4edn: Pearson.

Deshpande, A., Guestrin, C., Madden, S.R., Hellerstein, J.M., & Hong, W. (2004). Model-driven data acquisition in sensor networks. In *VLDB*.

Ge, T., Zdonik, S., & Madden, S. (2009). Top-*k* queries on uncertain data: On score distribution and typical answeres. In *SIGMOD*.

Hua, M., Pei, J., Zhang, W., & Lin, X. (2008). Ranking queries on uncertain data: A probabilistic threshold approach. In *SIGMOD*.

Ilyas, I.F., Beskales, G., & Soliman, M.A. (2008). *A survey of top-k query processing techniques in relational database systems*: ACM Computing Surveys.

Jestes, J., Cormode, G., Li, F., & Yi, K. (2011). *Semantics of ranking queries for probabilistic data*: TKDE.

Kanagal, B., & Deshpande, A. (2008). Online filtering, smoothing and probabilitic modeling of streaming data. In *ICDE*.

Li, J., & Deshpande, A. (2010). Ranking continuous probabilistic datasets. In *VLDB*.

Li, J., Saha, B., & Deshpande, A. (2009). A unified approach to ranking in probabilistic databases. In *VLDB*.

Lian, X., & Chen, L. (2008). Probabilisitc ranked queries in uncertain databases. In *EDBT*.

Lian, X., & Chen, L. (2011). *Probabilistic inverse ranking queries in uncertain databases*: The VLDB Journal.

Lyness, J.N. (1969). *Notes on the adaptive simpson quadrature routine*: Journal of ACM.

Pei, J., Hua, M., Tao, Y., & Lin, X. (2008). Query answering techniques on uncertain and probabilistic data. In *SIGMOD*.

Ré, C., Dalvi, N., & Suciu, D. (2007). Efficient top-k query evaluation on probabilistic data. In *ICDE*.

Sarma, A.D., Benjelloun, O., Halevy, A., & Widom, J. (2006). *Working models for uncertain data*.

Soliman, M.A., & Ilyab, I.F. (2008). *Probabilistic top-k and ranking-aggregate queries*: TODS.

Soliman, M.A., & Ilyas, I.F. (2009). Ranking with uncertain scores. In *ICDE*.

Soliman, M.A., Ilyas, I.F., & Chang, K.C.C. (2007). Top-*k* query processing in uncertain databases. In *ICDE*.

Song, C., Li, Z., & Ge, T. (2013). Top-k oracle: A new way to present top-k tuples for uncertain data. In *ICDE*.

Tao, Y., Cheng, R., Xiao, X., Ngai, W.K., Kao, B., & Prabhakar, S. (2005). Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *VLDB*.

Wang, C., Yuan, L.Y., You, H.H., & Zaiane, O.R. (2011). On pruning for top-k ranking in uncertain databases. In *VLDB*.

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