Journal of Intelligent & Fuzzy Systems 31 (2016) 1109–1120
DOI:10.3233/JIFS-169040
IOS Press

Probabilistic top-*k* range query processing for uncertain databases

Guoqing Xiao[a], Fan Wu[a,*], Xu Zhou[a] and Keqin Li[a,b]
[a]*College of Information Science and Engineering, Hunan University, Changsha, Hunan, China*
[b]*Department of Computer Science, State University of New York, New Paltz, NY, USA*

Abstract. Query processing over uncertain data is very important in many applications due to the existence of uncertainty in real-world data. In this paper, we propose a novel and important query for uncertain data, namely probabilistic top-(k, l) range (PTR) query, which retrieves l uncertain tuples that are expected to meet score range constraint [s_1, s_2] and have the maximum top-k probabilities but no less than a given probability threshold q. In order to accelerate the PTR query, we present some effective pruning techniques to reduce the search space of PTR query, which are integrated seamlessly into an efficient PTR query procedure. Extensive experiments over both real-world and synthetic datasets verify the efficiency and effectiveness of our proposed approaches.

Keywords: Data management, probabilistic top-*k* query, query processing, range query, uncertain data

1. Introduction

Uncertain data exists widely in information retrieval, mobile object tracking, web services and other various applications. In particular, some large-scale applications, for instance, sensor networks and RFID, can generate more uncertain data. There are many reasons for producing uncertain data, such as data noise, data leakage, transmission delay and inaccuracy or incompletion in measurement, etc. Similarly, surveys and imputation techniques crate data which is uncertain in nature. This has created a need for uncertain data management algorithms and applications [2, 18], among which a pivotal technique in this respect is the query processing over uncertain database, such as top-*k* query [6, 9, 12–14, 17, 20]. With the rapid development of data collection methods and the practical applications, the issue of uncertain data query has drawn large amounts of attention in both academia and industry [3].

*Corresponding author. Fan Wu, College of Information Science and Engineering, Hunan University, Changsha 410082, Hunan, China. Tel.: +86 18907483881; Fax: +86 0731 88664161; E-mail: wufan_whu@163.com.

Top-*k* query is a classic problem in the area of information retrieval. For a user-defined scoring function and one given query, algorithms return *k* objects which have the largest scores. However, in uncertain data management, data records are typically represented by probability distributions rather than deterministic values [2]. Therefore, traditional definite top-*k* query cannot respond to uncertain top-*k* query, thereby we have to redefine query semantics of top-*k* for uncertain query. As for uncertain top-*k* query, the interaction between score and probability determine the answers. Different combination of the two factors may generate various uncertain query semantics, among which most are based on possible worlds semantics [7]. In this paper, we propose an extended uncertain top-*k* query top-(k,l) query and several effective pruning strategies on the basis of a given probability threshold. It will be introduced in detail in Sections 3 and 5.

The range-based uncertain data query processing problem has drawn increasing attention to many researchers, and emerged some range-based uncertain query algorithms, such as location-based query [4], range-based nearest neighbor (NN) query [11] and skyline range query [15]. To the best of our knowledge, very few works refer to uncertain top-k range query processing. In this paper, we propose a range-based probabilistic top-(k,l) query (PTR query), i.e., for a given probability threshold q and a score range constraint $s = [s_1, s_2]$, the algorithm returns l tuples which meet s and their top-k probabilities are no less than q. We will introduce it at length in Sections 3 and 5.

Our contributions made in this paper are as follows.

- We first develop an new and crucial query, i.e., probabilistic top-(k,l) range (PTR) query by taking range query and top-k query into overall consideration in the context of uncertain databases.
- We present several effective pruning rules to reduce the search space, which are integrated into an efficient PTR query procedure.
- Particularly, we can find that a lower bound of top-k probability of certain tuple and an upper bound of the answer set by studying the properties of top-k probability.
- Extensive experiments are conducted over both real-world and synthetic data to evaluate the performance of the proposed algorithm. The experimental results show that our query algorithm perform well.

The rest of this paper is organized as follows. Section 2 reviews related work on uncertain top-k queries and range-based queries. Section 3 mainly introduces the uncertain data model and further gives the formal definitions concerning PTR query processing. Section 4 presents some properties of top-k probability and corresponding pruning rules. Section 5 pictures the PTR query algorithm. Section 6 evaluates the proposed algorithm with experiments. Finally, we make a conclusion with directions for future work.

2. Related work

We review the existing works on uncertain (probabilistic) top-k query and range-based query in this section.

2.1. Uncertain Top-k query processing

While research works on conventional top-k queries are mostly based on some deterministic scoring functions, the new factor of tuple membership probability in uncertain database makes evaluation of probabilistic top-k queries very complicated since the top-k answer set depends not only on the ranking scores of candidate tuples but also their probabilities. As for uncertain databases, there exist all kinds of uncertain top-k query semantics, among which the most influential include U-Top$k$ [17], U-kRanks [17], PT-k [12], Global-Top$k$ [20], Expected Rank [6], E-Score Rank [6], PTkS [14], and PTI query [19] etc. U-Top$k$ returns the most probable k-tuple vector with the maximum aggregated probability of being top-k over all possible worlds. U-kRanks returns a list of k tuples such that the i^{th}-ranked tuple has the highest aggregated probability in all possible worlds. These two query algorithm proposed in [17] are of inefficient due to lacking of pruning rules with an increasing possible world space. PT-k query returns those tuples which their top-k probabilities across all possible worlds are no less than a given probabilistic threshold q, which makes for approving performance without unfolding all possible worlds. Furthermore, a sampling method is developed to quickly compute an approximation with quality guarantee to the answer set by extracting a small sample of the uncertain dataset. Though the sampling method can lower the accuracy of answers, it can improve efficiency to a large degree. In [20], Zhang etc propose a Global-Top$k$ query semantic that returns k highest-ranked tuples on the basis of their probabilities of being top-k answer set in all possible worlds. In [6], the expected rank of each tuple over all possible worlds is regarded as the ranking function for obtaining the answer set. E-Score Rank query takes the E-Score of each tuple as the ranking function to find the final answers. Lian and Chen [14] propose the probabilistic top-k star (PTkS) query, which aims to retrieve k objects in an uncertain database that are "closest" to a static/dynamic query point, considering both distance and probability aspects. In [18], the authors present the problem of Top-K frequent itemsets mining in sliding windows. Xiao et al. proposed a probabilistic top-l influential (PTI) query to identify the l most favorite objects [19]. From these definitions, we can see that a pivotal problem of uncertain top-k query is the calculation of possible world probability. Provided there is no good pruning technology, the performance of query may be comparative low with an exponential growth large possible world space. Consequently, it is necessary to exploit some efficient pruning strategies to reduce the computing of top-k probability for improving the performance of algorithm.

2.2. Range-based query processing

Range-based query processing has recently more and more attention in various practical applications as a result of the uncertainty, such as moving object tracking [10], location-based services [15] and computer games [21] etc. For the uncertainty of object, or privacy reasons, the records we want to query usually locate in a finite range region, such as an interval range, rectangular or circular range. Existing some rage-based query algorithms, e.g., range-based *k*NN query [11], range-based skyline query [15] etc. In [11], Hu and Lee firstly presented the R*k*NN solution for rectangular ranges. Lin et al. proposed the first range-based skyline query in LBS [15]. In [5], Cheng et al. firstly put forward probabilistic range query based on one-dimensional space. Tao et al. studied the uncertain range query for arbitrary probability distribution function in multi-dimensional space. To the best of our knowledge, there is very little works that has studied range-based uncertain top-*k* queries. Brodal [1] and Tao [16] developed a static and dynamic structure for the top-*k* range reporting problem, respectively. In this paper, we propose an uncertain top-*k* range query based on score attribute range, which retrievals the uncertain database by appointing a score range.

3. Preliminaries

In this section, we first introduce an uncertain data model, then we give the formal definitions concerning the probabilistic top-(*k*,*l*) range query.

3.1. Uncertain date model

The fundamental difference between a traditional deterministic database and an uncertain database is that an uncertain relation represents a set of possible relation instances, rather than a single one [2]. Suppose an uncertain database DB, which is composed of a set of n tuples $t_i\,(1 \le i \le n)$. The uncertainty of every tuple t_i in the uncertain database DB is mainly represented by a confidence, i.e., its existence probability $P(t_i)$ in DB. For a given score ranking function $s(t)$, the score of each t_i is denoted by $s(t_i)$. In fact, many uncertain data processing, including top-*k* query processing, pay attention to mainly two types of uncertainty, i.e., tuple-level uncertainty and attribute-level uncertainty. For tuple-level uncertainty, every tuple is uncertain while its attribute value (i.e. score) is deterministic. For attribute-level uncertainty, every tuple is deterministic while its attribute value is uncertain, and every attribute value corresponds to a probability. In a probabilistic database, there may exist some generation rules between tuples, such as exclusion or coexistence, but almost tuples are independent of each other. In this paper, we only consider tuple-level uncertainty with all tuples mutually independent.

There exist many works on modeling uncertain data. One of the most popular is the model based on possible world semantics [7], where an uncertain database is regarded as a set of possible world instances associated with their probabilities. Each possible world W is a subset of uncertain database tuples, and the set of all worlds is denoted by the possible world space Ω. The probability of each world is computed as the joint probability of the existence of the world's tuples and the absence of all other database tuples. Since all tuples are mutually independent, we can obtain

$$P(W) = \prod_{t \in W} P(t) \prod_{t \notin W} \bar{P}(t), \tag{1}$$

where $\bar{P}(t) = 1 - P(t)$, and $\sum_{W \in \Omega} P(W) = 1$.

3.2. Problem definition

With the aforementioned introductions and possible world semantics, in this subsection, we put forward the query semantics concerning probabilistic top-(*k*,*l*) query based on a given probability threshold *q* and Refs. [9, 12, 20].

Definition 1. (Score dominating) Let $s(t)$ be a score ranking function, for arbitrary two tuples t_i, t_j, if $s(t_i) > s(t_j)$, then $t_i \succ_s t_j$.

Definition 2. (Top-k probability) Let DB be an uncertain database with possible world space Ω, k be a positive integer, $s(t)$ be a score ranking function, and $\text{Top}_k(W)$ be a set of k tuples in the front of possible world W on the basis of scoring function $s(t)$. Then the probability of any tuple t in DB $P_{top-k}(t, DB)$ can be defined as the summation of the probabilities of all possible worlds whose *top-k* answer set $\text{Top}_k(W)$ contains t , i.e.,

$$P_{top-k}(t, DB) = \sum_{W \in \Omega,\, t \in Top_k(W)} P(W). \tag{2}$$

Note that top-k answer sets may be of cardinality less than k for some possible worlds. We call such possible worlds as small worlds [20].

Definition 3. (Probabilistic top-(k, l) range query, PTR query) Let DB be an uncertain database, l be a positive integral number, s be a given score range constraint (e.g., $s = [s_1, s_2]$), q be a probability threshold, and $P_{top-k}(t, DB)$ be the top-k probability of tuple t in DB. Then the top-(k,l) query over uncertain database DB returns l tuples which meet constraint s and have the maximum top-k probabilities but no less than q, i.e.,

$$\{t|s(t) \in s, P_{top-k}(t, DB) \geq q\}, \tag{3}$$

and

$$|\{t|s(t) \in s, P_{top-k}(t, DB) \geq q\}| = l. \tag{4}$$

4. Pruning rules

On the basis of semantics of uncertain top-(k,l) query, we can know that the calculation of top-k is very large with exponentially growing possible worlds. Therefore, it is necessary to put forward some pruning techniques for reducing the computing of top-k.

Lemma 1. (*Pruning Rule 1*) *Let $s(t)$ be a score ranking function, s be a given score range constraint, DB be an uncertain database. For any tuple t, if $s(t) \notin s$, then we can remove t immediately from the DB without calculating its top-k probability.* □

Theorem 1. *The top-k probability of tuple t in an uncertain database DB, denoted by $P_{top-k}(t, DB)$, is at most its presence probability $P(t)$, i.e., $P_{top-k}(t, DB) \leq P(t)$.*

Lemma 2. (*Pruning Rule 2*) *Let q be a probability threshold, $P(t)$ be the presence probability of tuple t, $P_{top-k}(t, DB)$ be t's top-k probability. If $P(t) < q$, then t can be excluded immediately from the DB without computing $P_{top-k}(t, DB)$.*

According to Lemmas 1 and 2, before the calculation of top-k probability of a tuple, we can pre-prune some tuples potentially based on a given score range constraint s and a probability threshold q, respectively. They aim at decreasing the size of the uncertain dataset.

Lemma 3. (*Pruning Rule 3*) *Given an uncertain database DB with cardinality n, and $t_1 \succ_s t_2 \succ_s \cdots \succ_s t_n$. Let DB_{t_i} be the subset $\{t_1, t_2, \ldots, t_i\}$, $P(DB_{t_i}, j)$ denote the probability of any j tuples appearing in the set DB_{t_i}, denoted by $x_{t_i,j}$. If $\sum_{j=0}^{k-1} x_{t_i,j} < q$, then we can prune the tuples which rank lower than t_i, i.e., for $\forall\, 1 \leq m \leq n - i, m \in N_+$, $P_{top-k}(t_{i+m}, DB) < q$, where q is a given probability threshold.*

According to the pruning technique 3, we can obtain a compact set about the answer set. In subsequent experiments, we show that the pruning technology has excellent efficiency and effectiveness with pruning rate at least 99%.

The three pruning techniques aforementioned all aim at cutting down the size of the uncertain dataset for reducing the query search space. However, these pruning strategies do not or few consider the properties of top-k probability of tuples. One important innovation of this paper is the demonstration of the mathematical property of top-k probability of tuples. Next, we will give out the other pruning rules based on the nature.

Theorem 2. *Given an uncertain database DB with $t_1 \succ_s t_2 \succ_s \cdots \succ_s t_n$. Then $P_{top-k}(t_i, DB)$ increases with k, but no more than $P(t_i)$.*

Lemma 4. (*Pruning Rule 4*) *Given a probability threshold q, an uncertain database DB with cardinality n, and $t_1 \succ_s t_2 \succ_s \cdots \succ_s t_n$. Then there exists a lower bound LB of top-k probability of tuple t_i, i.e., $P(t_i) \prod_{t_j \succ_s t_i}(1 - P(t_j))$, such that we can add t_i into the top-(k,l) answer set without computing its complete top-k probability when $LB \geq q$.*

As a matter of fact, the LB of tuple t is its skyline probability in this uncertain DB if we take the uncertain data as the points in two-dimensional space [8, 22, 23].

Note that, we may put directly the tuples whose lower bound LB are equal or greater than q into the answer set. So the number of tuples in top-(k,l) answer set of the PTR query may larger than l on account of adding some tuples whose LB are no less than q. In subsequent experiments, we will demonstrate that the probability lower pruning not only reduces the PTR query search space, but it also grasps some relatively important tuples missed by some uncertain top-k queries, e.g., PT-k query and Global-Top$k$ query.

In summary, we can decrease efficiently the search space by removing some tuples based on these pruning rules we proposed, such as the score range constraint pruning (Pruning Rule 1), the probability threshold pruning (Pruning Rule 2), the upper bound of answer set pruning (Pruning Rule 3) and the top-*k* probability lower bound pruning (Pruning Rule 4).

5. PTR query

In this section, we will present our probabilistic top-(*k*,*l*) range query processing algorithm, i.e., PTR query, including a description of the algorithm (Section 5.1) and complexity analysis of the algorithm (Section 5.2).

5.1. The algorithm framework

We now describe the probabilistic top-(*k*,*l*) range (i.e., PTR) query processing algorithm. The algorithm is presented in Algorithm 1. Initially, we empty a queue Q for saving the answer set (line 1). Clearly, one straightforward way to answer the uncertain query is in terms of its semantics and pruning techniques we proposed. That is, for each uncertain tuple in the uncertain database DB, we firstly remove some tuples which do not meet the score range constraint and probability upper bound constraint (i.e., probability threshold), based on Lemmas 1 and 2 (lines 2–6). After that, we can obtain a compact database with smaller cardinality. Then, we sort the database in accordance with the decreasing order based on the scoring function $s(t)$ and sequentially scan tuples in the database (line 7). Next, we further prune the dataset in light of the upper bound of top-(*k*,*l*) answer set (lines 8–10). Then for the remaining tuples in DB, we compute the lower bound of their top-*k* probability and insert the tuples whose LB are no less than q into answer set Q (lines 11–14). After that, for the remaining tuples in the uncertain database, we compute their probabilities of being top-*k* (line 15). Last but not least, on the basis of the top-*k* probabilities and q, we select l tuples which have the maximum top-*k* probabilities but no less than q, and put them into the answer set Q (line 16). Finally, the algorithm returns l as its output. Its correctness and complexity analysis are demonstrated in the following subsection.

Algorithm 1. PTR Query Processing

```
Input:
    an uncertain database DB with the cardinality $n$, an score ranking function $s(t)$, a probability threshold $q$, a score range $[s_1,s_2]$, two integral numbers $k$ and $l$
Output:
    top-($k$,$l$) query answer set $Q$
 1: initialize the top-($k$,$l$) answer set $Q \leftarrow \emptyset$;
 2: for each tuple $t$ in DB do
 3:     if ($s(t) \notin s$ or $P(t) < q$) then
 4:         remove $t$ from the database DB;
 5: sort for remaining tuples in DB in the decreasing order of the scoring function $s(t)$;
 6: if $\sum_{j=0}^{k-1} x_{t_i,j} < q$ then
 7:     prune the tuples which rank lower than $t_i$ from the DB;
 8: for the remaining tuples in DB, compute the lower bound LB of top-$k$ probability of tuple $t_i$ in DB;
 9: if $LB \geq q$ then
10:     put $t_i$ into top-($k$,$l$) answer set $Q$;
11: for the remaining tuples in DB, compute top-$k$ probability of tuple $t$ $P_{top-k}(t, DB)$;
12: select $l$ tuples which have the maximum top-$k$ probabilities but no less than $q$, and put them into the answer set $Q$;
13: return $Q$.
```

5.2. Correctness and complexity analysis

In this section, we will concentrate on the proof of correctness of the PTR query processing algorithm and the analysis of complexity.

Theorem 3. *The algorithm of top-(k,l) range query processing, i.e., PTR query processing, on the basis of pruning rules we proposed can accurately find out the optimal tuples we wanted.*

Proof. From the definition of top-(*k*,*l*) range query, we can know that, for a score range constraint s and a probability threshold q, the query returns l tuples which meet constraint s and have maximum *top-k* probabilities but no less than q, i.e.,

$$\{t|s(t) \in s, P_{top-k}(t, DB) \geq q\}, \tag{5}$$

and

$$|\{t|s(t) \in s, P_{top-k}(t, DB) \geq q\}| = l. \tag{6}$$

Note that the top-(*k*,*l*) range query processing functions only on remaining dataset after Pruning Rules 1, 2 and 3, so the three pruning techniques do not have an effect on the accuracy of the query. On the other hand, as previously mentioned, the number of PTR query answer set may larger than l on account of adding some tuples whose LB are no less than q. But however it happens, the answer set of the PTR query always reports the optimal tuples we wanted.

Thus, as long as Line 11 of Algorithm 1, correctly calculates the top-*k* probability of tuples, the query can return a valid top-(*k*,*l*) answer set. Based on the Proposition 1 in Ref. [4], we can know that Algorithm 2 accurately compute the probability of being top-*k* of tuples in Equation (4). □

Theorem 4. *The complexity of PTR query processing algorithm, in the worst case, is* $max[O(n), O(n_1 * n_1), O(kn_2), O(n_2 log n_2)]$, *where,* n, n_1, n_2 *are the cardinality of original uncertain database, compact database based on score value constraint pruning and probability threshold pruning, and dataset on the grounds of the upper bound of answer set pruning and top-k probability lower bound pruning, respectively.*

Proof. Now we consider the complexity of the PTR query processing algorithm in the following. The most complicated operations are sorting the uncertain database, the upper bound of answer set pruning and the calculation of probability of being top-k of tuples. Assume that the cardinality of original indeterminate database is n, then the complexity of Pruning Rules 1 and 2 based on score range constraint and probability threshold is $O(n)$. Next, we need to sort the remaining tuples in compact database, and the worst sorting operation is $O(n_1 * n_1)$, where n_1 is the cardinality of the compact database. Then, the upper bound of answer set pruning takes $O(n_1 * n_1)$ in the worst case to compute a dynamic programming table. The following lower bound pruning cost, in the worst case, is $O(n_1 * n_1/2)$. In practice, the cost is far from it because the LB value may be zero for the lower-ranked tuples. Suppose the number of tuples in database is n_2 this moment. Then, Algorithm 2 takes $O(kn_2)$ to calculate a dynamic programming table for the computing of top-k probability of tuples. Finally, the query algorithm takes $O(n_2 log n_2)$ in the worst case to maintain the sets Q. Consequently, the worst case time complexity of PTR query processing algorithm is $max[O(n), O(n_1 * n_1), O(kn_2), O(n_2 log n_2)]$. □

6. Experimental evaluation

In this section, we demonstrate the efficiency and effectiveness of PTR query processing algorithm based on some pruning techniques we proposed through a series of simulations over both real-world and synthetic data. We take query execution time and PR (pruning rate, as defined hereinafter) as the primary performance metrics under various parameter settings. All experiments were run on a PC with a 2.60 GHz AMD Athlon^{TM} II X4 620 processor with 4 GB of main memory, and a 500 GB hard disk, running Windows Win7 32 bit Operating System. Our algorithm was implemented in Microsoft Visual Studio 2010.

Definition 4. (Pruning Ratio, PR) Let NUM_{bef} be the number of dataset before pruning, NUM_{aft} be the size of the dataset after pruning, we call

$$PR = \frac{NUM_{bef} - NUM_{aft}}{NUM_{bef}} \tag{7}$$

pruning ratio, denoted as PR. It represents the efficiency of pruning techniques.

6.1. Results on the real-world dataset

We use the International Ice Patrol (IIP) Iceberg Sightings Database[1] to evaluate the effectiveness of PTR query over uncertain data in real-world applications. This dataset was used in previous works on ranking queries in uncertain data [12]. The IIP collects information on iceberg activity in the North Atlantic. Its mission is to measure, plot and predict iceberg drift, and broadcasting all known ice to prevent icebergs threatening. In this database, each sighting record is regarded as a tuple, including some of importance attributes, such as sighting source, position (longitude and latitude) and the number of the drifted days etc. Among them, the number of days of iceberg drift is dated from the IIP drift and deterioration computer model, which is pivotal in determining the status of icebergs. It is interesting to find the icebergs drifting for a long period. In this real experiment, we consider drifted days as attribute score of tuples for top-k queries, and the larger the score value is, the more important the tuple is.

Moreover, each sighting record in this database is associated with a confidence-level in terms of the source of sighting. There are six types of sighting source, including 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). The differences in confidence-level of these sighting sources are classified and presented by probability values which are 0.8, 0.7, 0.6, 0.5, 0.4 and 0.3 to the six confidence-levels aforementioned, respectively.

The IIP_2009 database which contains 13,095 tuples is used to conduct the real experiment. We apply our PTR query and two other queries, i.e., PT-k query and Global-Top$k$ query, on the uncertain dataset. The ranking order is the number of days of iceberg drift descending order. For PTR query, we set $k = l = 10$, $q = 0.3$ and $s = [100, 500]$, that is, the researchers would want to observe those records

[1]http://nsidc.org/data/g00807.html

Table 1
PTR query answer set over the IIP_2009 database

Tuples	$\mathbf{t_{6903}}$	$\mathbf{t_{3610}}$	t_{3612}	t_{6928}	t_{4174}	t_{4020}	t_{8313}	t_{8412}	t_{8411}	t_{8409}	t_{8410}	t_{8408}
Drifted days	**500**	**500**	495	488.7	439.5	427.6	423.5	455.5	435.2	431.6	431	430.9
Presence prob.	**0.8**	**0.6**	0.8	0.8	0.8	0.8	0.8	0.7	0.7	0.7	0.7	0.7
Top-10 prob.	**–**	**–**	0.8	0.8	0.8	0.8	0.8	0.7	0.7	0.7	0.7	0.7
Rank	**1**	**2**	3	4	6	11	12	5	7	8	9	10

Table 2
PT-*k*/Global-Top*k* queries answer set over the IIP_2009 database

Tuples	t_{6903}	t_{3612}	t_{6928}	t_{4174}	t_{4020}	t_{8412}	t_{8411}	t_{8409}	t_{8410}	t_{8408}	$\mathbf{t_{3610}}$
Drifted days	500	495	488.7	439.5	427.6	455.5	435.2	431.6	431	430.9	**500**
Presence prob.	0.8	0.8	0.8	0.8	0.8	0.7	0.7	0.7	0.7	0.7	**0.6**
Top-10 prob.	0.8	0.8	0.8	0.8	0.766956	0.7	0.7	0.7	0.7	0.7	**0.6**
Rank	1	3	4	6	11	5	7	8	9	10	**2**

Table 3
The experimental datasets under various distributions

Datasets	Meanings
uu	the score and probability follow uniform distribution;
un(0.5)	the score follow uniform distribution, and the probability follow normal distribution with mean value equals to 0.5;
un(0.9)	the score follow uniform distribution, and the probability follow normal distribution with mean value equals to 0.9;
uexp(0.2)	the score follow uniform distribution, and the probability follow exponent distribution with mean value equals to 0.2;
uexp(0.5)	the score follow uniform distribution, and the probability follow exponent distribution with mean value equals to 0.5.

whose drifted days are located at between 100 and 500, and their probabilities of being top-10 are no less than 0.3. We set $k = 10$ and $q = 0.3$ for PT-k query and $k = 10$ for Global-Top$k$ query. The detailed information about the result sets of these three queries are showed in Tables 2 and 3, including attribute scores (drifted days), presence probabilities, top-10 probabilities and the corresponding ranks.

All tuples with top-10 probability at least 0.3 and drifted days between 100 and 500 are reported by the PTR query. As is shown in Table 1, the PTR query returns a set of 12 tuples $\{t_{6903}, t_{3610}, t_{3612}, t_{6928}, t_{4174}, t_{4020}, t_{8313}, t_{8412}, t_{8411}, t_{8409}, t_{8410}, t_{8408}\}$ as the top-(10,10) answer set, among which t_{6903} and t_{3610} (as shown in bold) are obtained by probability lower bound pruning (Pruning Rule 4). Table 2 illustrates the results of PT-k query and Global-Top$k$ query. Although tuple t_{3601} (as shown in bold) has the relatively low presence probability, it is regarded as a member of the answer set for the reason that its score (drifted days) is larger or equal than that of tuples in the result set. However, the tuple t_{3601} is not included in the answer set of the PT-k query and Global-Top$k$ query despite the fact that t_{3601} has the largest score in the given score range. In other words, the two queries lose some relatively important tuples. From an real experiment viewpoint, it verifies also the effectiveness of PTR query, that is, the query can accurately report those optimal tuples users wanted.

The pruning ratios of the score range pruning and probability threshold pruning (i.e., pruning rules 1 and 2), the upper bound pruning of answer set (i.e., pruning technique 3) and probability lower bound pruning (i.e., pruning strategy 4) are 78.68%, 99.46% and 13.33%, respectively. The upper bound pruning of result set has a super pruning effect with the pruning ratio at least 99%. The effects of pruning of pruning techniques 1 and 2 rests with the score range and probability threshold user selected. Although the pruning rule 4 has a relatively lower pruning effect, it contributes to our PTR query capturing those important tuples missed by some uncertain top-k queries such as the PT-k query and Global-Top$k$ query.

Furthermore, for the real-world uncertain database, we also conduct several experiments to evaluate the efficiency and effectiveness of PTR query under different parameter settings. The experimental results are shown in Figs. 1–3. Figure 1 pictures the query running time of the query with k value up to $1K$ ($K = 1000$, similarly hereinafter), where $s = [100, 500]$,

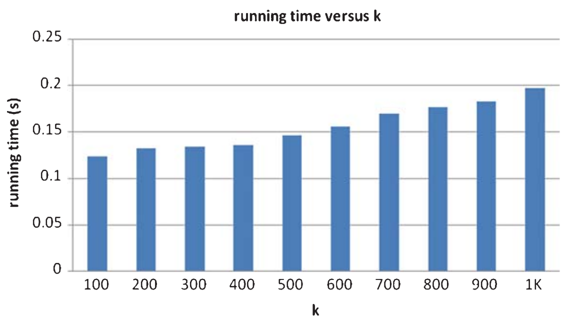

Fig. 1. Running time versus k over the IIP_2009 database.

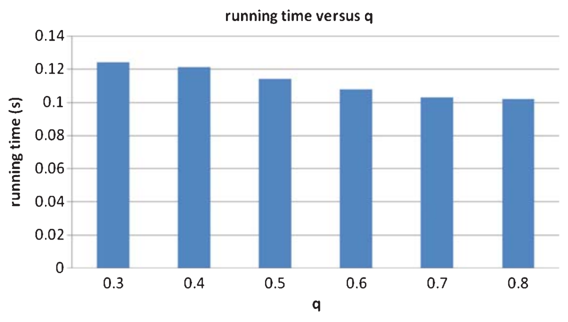

Fig. 2. Running time versus q over the IIP_2009 database.

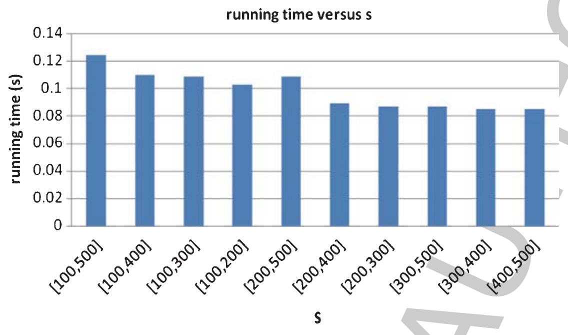

Fig. 3. Running time versus s over the IIP_2009 database.

$q = 0.3$ and $l = 100$. As illustrated in this picture, the time increases almost linearly with k value, which is intuitive. In the other one trial, we illustrate the influence of probability threshold on the efficiency and effectiveness of PTR query. Figure 2 describes the experimental results under the setting of $s = [100, 500]$ and $k = l = 100$. We can see intuitively from the picture that the execution time decreases as q value increases. However, the tendency of decreasing is not obvious for the reason that the size of IIP_2009 database is too small. In order to study the extendibility of PTR query, we will test the PTR query on massive synthetic data in the next subsection. In the last one experiment, we illustrate the effect of various score range s on the query time, where $q = 0.3$ and $k = l = 100$. The experimental result is described in Fig. 3. As shown, we can evaluate the performance of PTR query based on different score range users given. As a matter of fact, users can decide the range interval on the basis of their own preference.

In summary, our PTR query captures those important tuples missed by the PT-k query and Global-Top$k$ query with better effectiveness. This experiment on real-world database elaborates also the differences among the various kinds of top-k queries for uncertain data. In the following, we will test the PTR query on synthetic datasets for evaluating its scalability and performance further.

6.2. *Results on the synthetic dataset*

In order to evaluate the query answering quality and the scalability of our proposed algorithms, we generate some kinds of synthetic data. Since uncertain top-(k,l) query algorithm needs to balance scores and probabilities of tuples, the experimental data should consider different distribution with respect to the scores and probabilities. In this section, we conduct the experiments on some synthetic datasets under different distributions. Their scores all follow the uniform distribution between zero and one hundred, and their probabilities obey uniform distribution, normal distribution and exponent distribution, respectively. There is no correlation between the score and the probability. Table 3 illustrates the experimental datasets we adopted under various distributions.

We still take query running time as the primary performance metric under different parameter settings. Note that the time includes the sorting time after Pruning Rules 1 and 2.

6.2.1. *Effects of the cardinality of database n*

In this series of experiments, we vary the number of uncertain data n from $16K$ to $512K$. The best case for the query algorithm is to find highly probable tuples frequently after sorting the uncertain database, which allows obtaining strong candidates to prune other low probability candidates aggressively based on these pruning techniques, and thus report the query quickly. Figure 4 shows the execution time of PTR query with the cardinality of dataset n up to $512K$ under different datasets, where $s = [10, 90]$, $q = 0.2$ and $k = l = 100$. We can see from the picture the time increases sharply as n value increases, which is intuitive. For example, for the uniform distribution pair, the PTR query time approaches 2,500 seconds for the size of database of $512K$, while the time is under 1 seconds for $16K$. *Uu* distribution pair and *un* distribution pair have the relatively big running time while *uexp* distribution pair (e.p., the $exp(0.2)$ distribution pair) has the relatively small execution time for the PTR query. This can be explained based on the fact that a small quantity of tuples have relatively high probability under *uexp* distribution, thereby the number of prunable tuples increase with n on the basis of the pruning strategies. However, the *un* distribution pair where a considerable number of tuples are highly probable, leading to the prunable tuples decrease based on the pruning techniques. On the other hand, there is a relatively longer execution time when dataset meets *un* distribution with larger mean value, because their probabilities, especially some relatively high probabilities, distribute so intensively that we can prune less tuples. As such, there need a shorter time when dataset meets *uexp* distribution with smaller mean value. Because the mean value forces probability to decay relatively fast leading to a small number of highly likely tuples.

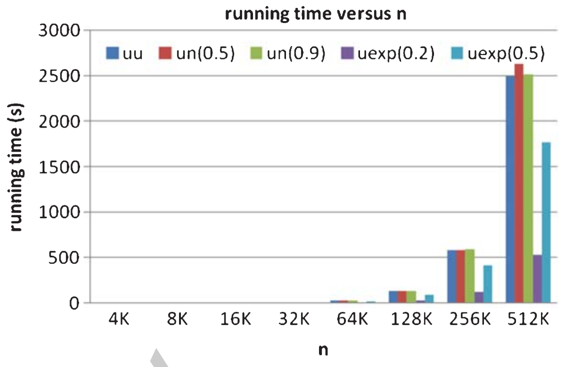

Fig. 4. Running time versus n (various probability distributions).

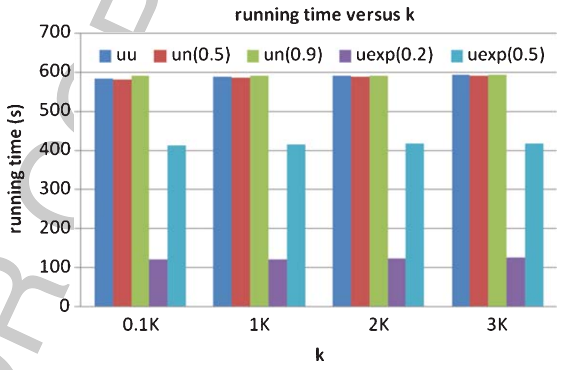

Fig. 5. Running time versus k (various probability distributions).

6.2.2. *Effects of the parameter k*

In this experiment, we study the influence of k value on the performance of PTR query under various types of datasets. We vary k value from $0.1K$ to $3K$. The experimental results are shown in Fig. 5. In this picture, we illustrates the query running time with k value up to $3K$, where $s = [10, 90]$, $q = 0.2$, $l = 100$ and $n = 256K$. From the picture, we can know that the execution time increases almost linearly as k value increases. PTR query execution time is close to 600 seconds for uniform distribution and norm distribution pair, while the running time is under 130 seconds for exponential distribution with mean value 0.2. The variation of running time of different distributions roughly has the similar form with the Fig. 5 as the same causes aforementioned.

6.2.3. *Effects of the probability threshold q*

In this train of experiments, we discuss the relationship between different probability threshold and performance of query. We vary probability threshold q value from 0.2 to 0.8. The experimental results are shown in Fig. 6. In this experiment, we illustrate the query running time with q value up to 0.8, where $s = [10, 90]$, $k = l = 100$ and $n = 256K$. From the picture, we can know that the query execution time decreases obviously as q value increases for the reason that the number of dataset decreases with q increases. For instance, the query time exceeds 590 seconds for threshold 0.2, while the time is under 35 seconds for threshold 0.8 under $un(0.9)$ distribution. On the other hand, as what mentioned before, the query time of various kinds of distribution datasets has the similar tendency of variation with Figs. 4 and 5.

6.2.4. *Effects of the score range s*

Now, we vary score rang constraint s for simulations. Figure 7 shows the experimental results. In the drawing, we demonstrate the PTR query execution time with score rang s, where $q = 0.2$, $k = l = 100$ and $n = 256K$. We describe the running time for different score range in the picture. Of course, in practice, the users may set the score range in terms of their own demands. On the other side, the variation of running time of various kinds of distribution datasets roughly has the approximate trend with the Figs. 4–6 as the similar causes mentioned previously.

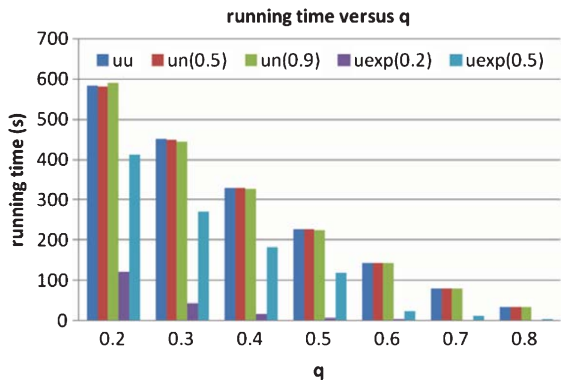

Fig. 6. Running time versus q (various probability distributions).

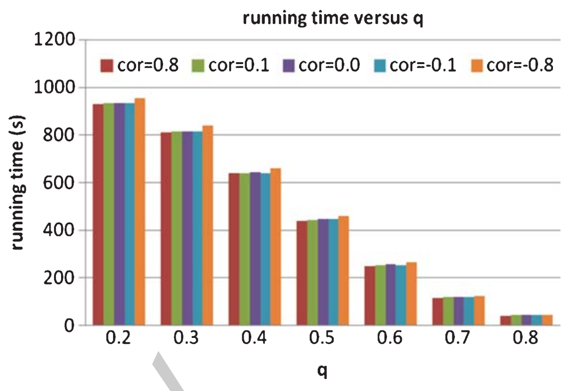

Fig. 9. Running time versus q (various bivariate normal distributions).

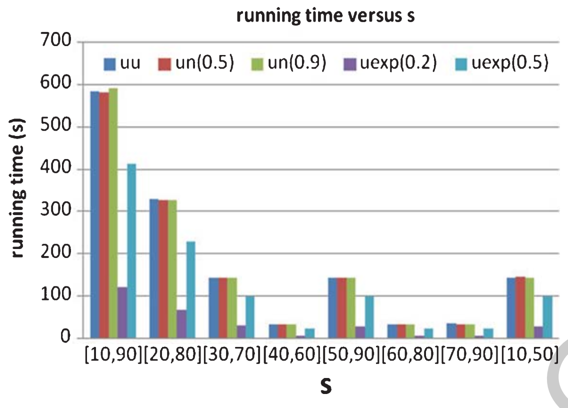

Fig. 7. Running time versus s (various probability distributions).

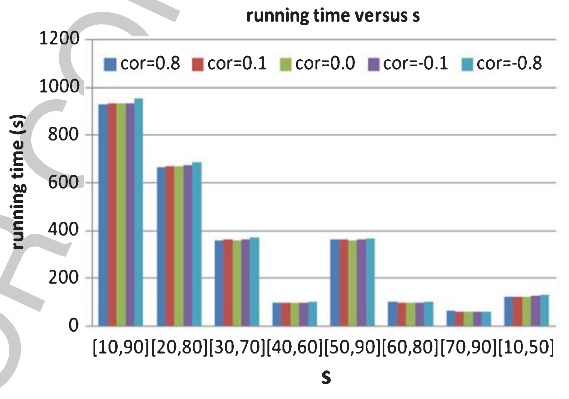

Fig. 10. Running time versus s (various bivariate normal distributions).

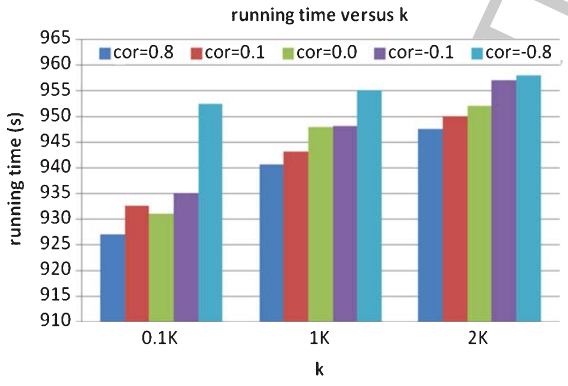

Fig. 8. Running time versus k (various bivariate normal distributions).

6.2.5. *Effects of the score-probability correlations*

In this series of experiments, we evaluate the effect of score-probability correlation on the performance of query. We use the synthetic uncertain databases of mutually independent tuples with the score and probability values of uncorrelation, positive correlation and negative correlation. Given this, we generate bivariate normal data over score and probability, and control the correlation coefficient by adjusting bivariate covariance matrix. The size of these synthetic dataset is all $256K$ with different correlation coefficient. We evaluate the performance of PTR query under various parameter settings. Experimental results are shown in Figs. 8–10. In Fig. 8, we study the effect of k value on the performance of PTR query under various correlation coefficient. We vary k value from $0.1K$ to $2K$, where $s = [10, 90]$, $q = 0.2$, $l = 100$ and $n = 256K$. Figure 9 illustrates the query execution time with q value up to 0.8, where $s = [10, 90]$, $k = l = 100$ and $n = 256K$. In addition, we discuss the influence of different score range on performance of query under the setting of $q = 0.2$, $k = l = 100$ and $n = 256K$, as illustrated in Fig. 10. We can know from these graphs that the positive correlation has positive effects on the performance of

Table 4
The pruning ratio under different probability distributions

Datasets/PR	PR_{12}	PR_3	PR_4
uu	35.28%	99.94%	8.20%
$un(0.5)$	35.38%	99.95%	8.09%
$un(0.9)$	35.31%	99.95%	6.25%
$uexp(0.2)$	45.80%	99.93%	5.56%
$uexp(0.5)$	70.29%	99.82%	2.94%

Table 5
The pruning ratio under various bivariate normal distributions

Datasets/PR	PR_{12}	PR_3	PR_4
dataset with cor=0.8	18.67%	99.92%	1.86%
dataset with cor=0.1	18.58%	99.92%	0.61%
dataset with cor=0.0	18.63%	99.92%	1.23%
dataset with cor=-0.1	18.55%	99.92%	1.19%
dataset with cor=-0.8	18.23%	99.92%	1.21%

query while anti-correlation has negative influences on the performance. This can be explained based on the fact that for the data of positive correlation, high score tuples are attributed with high probability, which allows pruning considerable low probable tuples in advance to answer uncertain top-k range queries, while for negatively correlated data, more tuples need to be visited before concluding results.

6.2.6. Pruning effects

In this subsection, we evaluate the efficiency and effectiveness of pruning techniques for different types of data used previously. Experimental results are shown in Tables 4 and 5. Table 4 illustrates the pruning ratios of pruning strategies under various kinds of probability distribution with the settings of $s = [10, 90]$, $q = 0.2$, $k = l = 100$ and $n = 512K$. Table 5 describes pruning ratios of these four pruning techniques under different bivariate gaussian data over score and probability with the settings of $s = [10, 90]$, $q = 0.2$, $k = l = 100$ and $n = 256K$. From the tables, we can see that the pruning rule 3 has an unexceptionable pruning efficiency with the pruning rate at least 99.9%. Pruning strategies 1 and 2, as the methods of predictive pruning, have also better pruning effects, which depends on the score range and probability threshold user selected. Although the pruning rule 4 has a lower pruning effect, it contributes to our PTR query capturing those important tuples missed by some uncertain top-k queries such as the PT-k query and Global-Top$k$ query.

In a word, extensive experiments based on synthetic data have very well verified the efficiency and effectiveness of our proposed algorithm for the uncertain top-k range query, in terms of few execution time excellent pruning effects and more optimal query results.

7. Conclusions

In real-world applications, uncertainty is inherently exist in data. It has recently become crucial to explore how to answer various queries for uncertain data effectively and efficiently. In this article, we develop an new and important probabilistic top-(k,l) range query, i.e., PTR query, algorithm in terms of a given attribute score range constraint and a probability threshold. In addition, we put forward several effective pruning strategies to improve the performance of the query algorithm we proposed. Extensive experiments have verified the efficiency and effectiveness of our proposed approaches. As for future work, we will extend the query processing algorithm on the basis of the assumptions aforementioned, including tuple-level uncertainty with any generation rules, attribute-level uncertainty.

Acknowledgments

The research was partially funded by the Key Program of National Natural Science Foundation of China (Grant Nos. 61133005, 61432005), the National Natural Science Foundation of China (Grant Nos. 61370095, 61472124), the International Science & Technology Cooperation Program of China (Grant No. 2015DFA11240), the National High-tech R&D Program of China (Grant No. 2015AA015303), the Hunan Provincial Innovation Foundation For Postgraduate (Grant No. CX2016B066), and the Outstanding Graduate Student Innovation Fund Program of Collaborative Innovation Center of High Performance Computing.

References

[1] P. Afshani, G. Brodal and N. Zeh, Ordered and unordered top-k range reporting in large data sets, *In Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SIAM, 2011, pp. 390–400.
[2] C.C. Aggarwal and P.S. Yu, A survey of uncertain data algorithms and applications, *IEEE Trans on Knowledge and Data Engineering* **21**(5) (2009), 609–623.
[3] I. Barba, B. Weber, C. Del Valle and A. Jiménez-Ramírez, User recommendations for the optimized execution of busi-

ness processes, *Data & Knowledge Engineering* **86** (2013), 61–84.

[4] J.C. Chen and R. Cheng, Efficient evaluation of imprecise location-dependent queries, *In Proc of the 23rd Int Conf on Data Engineering (ICDE)*, IEEE, 2007, pp. 586–595.

[5] R. Cheng, Y.N. Xia, S. Prabhakar, R. Shah and J.S. Vitter, Efficient indexing methods for probabilistic threshold queries over uncertain data, *In Proceedings of the 13th International Conference on Very Large Data Bases-Volume 30*, VLDB Endowment, 2004, pp. 876–887.

[6] G. Cormode, F. Li and K. Yi, Semantics of ranking queries for probabilistic data and expected ranks, *In Proc of the 25th Int Conf on Data Engineering (ICDE)*, IEEE, 2009, pp. 305–316.

[7] A. Das Sarma, O. Benjelloun, A. Halevy and J. Widom, Working models for uncertain data, *In Proc of the 22nd Int Conf on Data Engineering (ICDE)*, IEEE, 2006, pp. 7–7.

[8] X.F. Ding and H. Jin, Efficient and progressive algorithms for distributed skyline queries over uncertain data, *IEEE Trans on Knowledge and Data Engineering* **24**(8) (2012), 1448–1462.

[9] T.J. Ge, S. Zdonik and S. Madden, Top-k queries on uncertain data: On score distribution and typical answers, *In Proc of the ACM Int Conf on Management of Data (SIGMOD)*, ACM, 2009, pp. 375–388.

[10] B. Gedik, K.L. Wu, P.S. Yu and L. Liu, Processing moving queries over moving objects using motionadaptive indexes, *IEEE Trans on Knowledge and Data Engineering* **18**(5) (2006), 651–668.

[11] H.B. Hu and D.L. Lee, Range nearest-neighbor query, *IEEE Trans on Knowledge and Data Engineering* **18**(1) (2006), 78–91.

[12] M. Hua, J. Pei, W.J. Zhang and X.M. Lin, Ranking queries on uncertain data: A probabilistic threshold approach, *In Proc of the ACM Int Conf on Management of Data (SIGMOD)*, ACM, 2008, pp. 673–686.

[13] X. Lian and L. Chen, Probabilistic ranked queries in uncertain databases, *In Proc of the 11th Int Conf on Extending Database Technology: Advances in Database Technology (EDBT)*, ACM, 2008, pp. 511–522.

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

[15] X. Lin, J.L. Xu and H.B. Hu, Range-based skyline queries in mobile environments, *IEEE Trans on Knowledge and Data Engineering* **25**(4) (2013), 835–849.

[16] C. Sheng and Y.F. Tao, Dynamic top-k range reporting in external memory, *In Proceedings of the 31st symposium on Principles of Database Systems*, ACM, 2012, pp. 121–130.

[17] M.A. Soliman, I.F. Ilyas and K.C.C. Chang, Topk query processing in uncertain databases, *In Proc of the 23rd Int Conf on Data Engineering (ICDE)*, IEEE, 2007, pp. 896–905.

[18] J. Van Hulse and T. Khoshgoftaar, Knowledge discovery from imbalanced and noisy data, *Data & Knowledge Engineering* **68**(12) (2009), 1513–1542.

[19] G.Q. Xiao, K.L. Li and K.Q. Li, Reporting l most favorite objects in uncertain databases with probabilistic reverse top-k queries, *In 2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, IEEE, 2015, pp. 1592–1599.

[20] X. Zhang and J. Chomicki, On the semantics and evaluation of top-k queries in probabilistic databases, *In Proc of the 29th Int Conf on Data Engineering Workshops (ICDEW)*, IEEE, 2008, pp. 556–563.

[21] Z.J. Zhang, Y. Yang, A. Tung and D. Papadias, Continuous k-means monitoring over moving objects, *IEEE Trans on Knowledge and Data Engineering* **20**(9) (2008), 1205–1216.

[22] X. Zhou, K.L. Li, Y.T. Zhou and K.Q. Li, Adaptive processing for distributed skyline queries over uncertain data, *IEEE Trans on Knowledge and Data Engineering* **28**(2) (2016), 371–384.

[23] X. Zhou, Y.T. Zhou, G.Q. Xiao, Y.F. Zeng and F. Zheng, Effective approach for an extended p-skyline query, *Journal of Intelligent & Fuzzy Systems* (2016), 1–10.