Efficient and Progressive Algorithms for Distributed Skyline Queries over Uncertain Data

Xiaofeng Ding, Hai Jin

Services Computing Technology and System Lab
Cluster and Grid Computing Lab
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
{xfding, hjin}@hust.edu.cn

Abstract— **The skyline operator has received considerable attention from the database community, due to its importance in many applications including multi-criteria decision making, preference answering, and so forth. In many applications where uncertain data are inherently exist, i.e., data collected from different sources in distributed locations are usually with imprecise measurements, and thus exhibit kind of uncertainty. Taking into account the network delay and economic cost associated with sharing and communicating large amounts of distributed data over an internet, an important problem in this scenario is to retrieve the global skyline tuples from all the distributed local sites with minimum communication cost. Based on the well known notation of the probabilistic skyline query over centralized uncertain data, in this paper, for the first time, we propose the notation of distributed skyline queries over uncertain data. Furthermore, two communication- and computation-efficient algorithms are proposed to retrieve the qualified skylines from distributed local sites. Extensive experiments have been conducted to verify the efficiency and the effectiveness of our algorithms with both the synthetic and real data sets.**

I. Introduction

With the fast development of computing infrastructures and easily available network services, data are increasingly stored and processed distributively [12, 17, 19, 30]. More and more applications collect data from distributed sites and compute query results based on the collective view of the data from all local sites.

In many application domains, due to the large amounts of data stored and the network delay incurred, as well as the economic cost associated with data sharing and communication [1], it is often very expensive to communicate the entire data set from each local site to the centralized data server for query processing. As a result, many query semantics in such applications that rarely need transmitting every piece of data in the local sites have been well studied to speed up the computation and communication efficiency [13, 14, 15, 16]. For example, a ranking query is implemented and only the top-*k* records (i.e. the *k* largest score according to a predefined function) are retrieved [15, 29]; a skyline query is processed to obtain those points that preferred according to a customer's preference [14].

However, none of the aforementioned works deal with distributed query over uncertain data, which has emerged as a major data type in many applications. Interestingly enough, many cases where data uncertainty inherently exists are distributed, e.g., distributed sensor networks with imprecise measurements [18], multiple geographically distant data sources for information integration based on fuzzy similarity scores [20]. Recently, Li *et al.* [1] has studied top-*k* queries in a communication-efficient manner on probabilistic data from multiple, distributed sites. On the other hand, as another important research direction in such a context, skyline queries, which focusing attention on the most preferable records, have not been studied before. It is not surprise that study distributed skyline queries on probabilistic data will quickly attract a lot of interests.

Distributed skyline computation over uncertain data has many important applications. For instance, consider the stock market application where customers may want to select good deals (transactions) for a particular stock over all the distributed stock exchange centres. A deal is recorded by two attributes (price, volume) where price is the average price per share in the deal and volume is the number of shares. In such scenario, before making trade decisions, a customer may want to know the top deals over all the distributed local sites. Generally speaking, deal *a* is better than another deal *b* if *a* involves a higher volume and is cheaper than those of *b*. Nevertheless, recording errors caused by systems or human beings may make failed deals be recorded successful, and vise versa; consequently each deal recorded in the database has a probability to be true. Therefore, a set of deals recorded in the database may be treated as a set of uncertain elements and some customers may only want to know "top" deals (skyline) among the entire deals over distributed sites; and thus we have to take the uncertainty of each deal into consideration. This is a scenario of distributed skyline queries over uncertain data.

Skyline operator is an important query type, lots of efforts have been conducted to speed up the query efficiency, from both the database and networking communities [2, 3, 4, 6, 7, 14, 17, 20, 21] recently. However, none of these works has addressed the problem in a distributed uncertain environment. We observe that uncertainty and distributed skyline query processing have been each studied quite extensively but separately [8, 9, 12], despite the fact that the two often arise concurrently in many applications. In this paper, for the first time, we study the problem of distributed skyline queries over uncertain data, and propose computation- and communication efficient processing algorithms.

As an overview, we propose an algorithm, called *distributed skyline over uncertain data* (DSUD), which computes the global skylines in a distributed, uncertain environment with economical bandwidth cost. Furthermore, an enhance version of DSUD algorithm (e-DSUD) is also proposed to further speedup the query efficiency and reduce the communication cost. The efficiency of e-DSUD stems from a highly optimized feedback mechanism, where the central server transmits to each local site precious information that prevents the delivery of a large number of non-skyline points. Beside low bandwidth consumption, both DSUD and e-DSUD algorithms achieve excellent progressiveness in outputting the final skyline points.

Our contributions. We study skyline queries for distributed probabilistic data. We design communication efficient algorithms for retrieving the skyline tuples with the global skyline probability larger than a threshold from distributed sites, with computation overhead also as a major consideration. In summary, our contributions are as follows:

- For the first time we give the formal definition of distributed skyline queries, in the context of uncertain data.
- We study a general framework to answer the DSUD query, and propose an effective feedback mechanism, which can be seamlessly integrated into our query procedure, to help reduce the communication cost and find the most promising candidates for the skyline query.
- We conduct experiments under different parameter settings to verify the effectiveness of our proposed feedback mechanisms as well as the efficiency of the corresponding query processing algorithms.

The rest of the paper is organized as follows. Section II reviews the previous work that is directly related to ours. Section III formally defines the problem and introduces the baseline approach, and Section IV explains the general framework for efficient distributed skyline query processing. Section V presents the detail DSUD algorithm and its enhanced version, which adopts an efficient feedback mechanism. Section VI introduces the implementation issues, and Section VII evaluates the proposed algorithms with experiments. Section VIII concludes the paper with directions for future work.

II. Related Work

A. Distributed Skyline Queries

In the literature, a number of interesting methods have been made to address distributed skyline retrieval [20, 22, 23, 24, 25]. Balke *et al.* [23] proposed the first distributed skyline algorithm, by considering the underlying relation is vertically partitioned between local severs, i.e., each server keeps only an attribute of the relation. In particular, it assumes that a d-dimensional database is vertically partitioned into d lists, where each list contains the values of a dimension in ascending order. Based on the rationale of the TA (Threshold Algorithm), their algorithm performs sorted accesses over the d lists in a round-robin manner, until a point p comes up in all of them. Up to now, those points unseen in any list can be pruned, as they must be dominated by p, thus cannot be in the final skyline set. In the contrast, Wu *et al.* [25] assumed a constrained horizontal partitioning determined by P2P overlay networks CAN [27]. Specifically, each local server is mapped to a rectangular region contained in the data space and is responsible for managing all and only the data points that fall into its scope. The constrained skyline retrieval they proposed aims at returning the skyline of the data points falling in a query range.

Recently, Wang *et al.* [14] developed an efficient algorithm for retrieving the skyline in P2P networks BATON [28]. Similar to the work in [25], the algorithm demands that the data of the peers are organized using an overlay network. Vlachou *et al.* [24] discussed distributed skyline computation in various subspaces. Their algorithm can be used to fetch skylines of the full space, but it incurs significantly higher network bandwidth consumptions. Finally, Zhu *et al.* [22] studied the distributed skyline query when the underlying data set is horizontally partitioned onto a set of local servers. Their algorithm can solve the problem with low bandwidth consumption. However, none of the above work can deal with the skyline queries over distributed data with uncertainty.

B. Probabilistic Skyline Queries

In the context of uncertain database, the probabilistic skyline query processing on uncertain data is firstly studied by Pei *et al.* [26]. Specifically, they define the probabilistic skyline query where each object contains discrete random instances. Effective pruning rules have been proposed to facilitate the search process, and bounding-pruning-refining processing framework is developed for efficient probabilistic skyline computation. Furthermore, based on the reverse skyline semantics and the uncertain data model, Lian and Chen [6] studied efficient and accurate query processing of reverse skyline query over uncertain data, considering both monochromatic and bichromatic cases.

Recently, Zhang *et al.* [2] investigated the problem of efficiently computing skyline against sliding windows over an uncertain data stream regarding given probability thresholds. They present a framework based on efficiently maintaining a candidate set. They also show that such a candidate set is the minimum information that needs to be kept to answer the skyline query. Furthermore, efficient techniques have been presented to process continuous queries. The above algorithms, however, are not applicable in a distributed environment where the network bandwidth has to be considered.

III. Preliminaries

Uncertainty data model and possible world semantics. We use D to denote an uncertain database, and D is composed of a set of N tuples t_i ($1 \le i \le N$). Each tuple t_i has a probability $P(t_i)$ ($0 < P(t_i) \le 1$) to occur, and v_{ij} ($1 \le j \le d$) denotes the j-th dimension value. The uncertainty data model is depicted in Fig.1. Furthermore, based on this uncertainty data model, a possible world W is instantiated by taking a set of tuples from the uncertain relation, that is, each tuple selects its existential states independent of other tuples. Generally speaking, the probability of W appears is:

$$P(W) = \prod_{t \in W} P(t) \times \prod_{t \notin W} (1 - P(t)) \tag{1}$$

Tuples	Value	Probability
t_1	$(v_{11}, v_{12}, \dots, v_{1d})$	$P(t_1)$
t_2	$(v_{21}, v_{22}, \dots, v_{2d})$	$P(t_2)$
t_3	$(v_{31}, v_{32}, \dots, v_{3d})$	$P(t_3)$
$\vdots$	$\vdots$	$\vdots$
t_N	$(v_{N1}, v_{N2}, \dots, v_{Nd})$	$P(t_N)$

Fig. 1. The uncertainty data model

If we denote the set of all possible worlds be Ω, then we always have $\sum_{W \in \Omega} P(W) = 1$. The example in Fig.2 illustrates the possible worlds for the uncertain database in a two dimensional spaces based on the uncertainty data model described in Fig.1.

Tuples	Value	Probability
t_1	(80, 96)	0.8
t_2	(85, 90)	0.6
t_3	(75, 95)	0.8

World W	Probability P(W)
$W_1 = \{\emptyset\}$	0.2×0.4×0.2 = 0.016
$W_2 = \{t_1\}$	0.8×0.4×0.2 = 0.064
$W_3 = \{t_2\}$	0.2×0.6×0.2 = 0.024
$W_4 = \{t_3\}$	0.2×0.4×0.8 = 0.064
$W_5 = \{t_1,t_2\}$	0.8×0.6×0.2 = 0.096
$W_6 = \{t_1,t_3\}$	0.8×0.4×0.8 = 0.256
$W_7 = \{t_2,t_3\}$	0.2×0.6×0.8 = 0.096
$W_8 = \{t_1,t_2,t_3\}$	0.8×0.6×0.8 = 0.384

Fig. 2. An example of possible worlds

A. Problem Definition

Traditionally, for two tuples t and s, we say t dominates s, denoted by $t \prec s$, if the values of t are not larger than that of s in all dimensions, and there exists at least one dimension where the value of t is smaller than s. Given a set of tuples, the skyline is composed of those tuples which are not dominated by any other tuples.

In the context of uncertain data, the skyline definition is quite different from the conventional one, that is to say, instead of clearly assigning a tuple with YES or NO to be in the skyline, whether a tuple t is in the skyline set or not is measured by its skyline probability $P_{sky}(t, D)$. For a possible world W, let $sky(W)$ be the set of elements in W that compose the skyline of W, then the skyline probability $P_{sky}(t, D)$ of a tuple t equals to the summation over all existential probabilities of those possible worlds where t is contained in their skyline set, that is:

$$P_{sky}(t, D) = \sum_{t \in sky(w), w \in \Omega} P(W) \tag{2}$$

For example in Fig.2, the skyline probability for t_1 is $P_{sky}(t_1) = P(W_2) + P(W_5) = 0.064 + 0.096 = 0.16$. Similarly, $P_{sky}(t_2) = P(W_3) + P(W_5) + P(W_7) + P(W_8) = 0.6$, and $P_{sky}(t_3) = P(W_4) + P(W_6) + P(W_7) + P(W_8) = 0.8$. Furthermore, for an uncertain tuple t with existential probability $P(t)$, the probability that t appears in uncertain database D equals to $P(t)$. Considering another uncertain tuple t' with existential probability $P(t')$, if t' dominates t, then the probability that t is dominated by t' equals to $P(t')$. Thus, the probability that t is not dominated by any other tuples is $\prod_{t' \in D, t' \prec t} (1 - P(t'))$. We conclude that the skyline probability of tuple t in equation (2) can be rewritten as:

$$P_{sky}(t, D) = \prod_{t' \in D, t' \prec t} (1 - P(t')) \times P(t) \tag{3}$$

Based on the above definition, the problem of distributed skyline retrieval over uncertain data (DSUD) can be defined as follow:

Definition 1 (Distributed Skyline in Uncertain Data) Given a set of m distributed sites $S = \{s_1, s_2, \dots, s_m\}$, each possessing an uncertain database D_i ($1 \le i \le m$) with size n_i, and a centralized server H where the query is processed. A distributed skyline over uncertain data would like to report at H those tuples with their global skyline probabilities not smaller than a given threshold q ($0 < q \le 1$). Based on the semantics of skyline definition over a centralized database and the independence of those local databases D_i, the global skyline probability of tuple t, denoted as $P_{g\text{-}sky}(t)$, is defined as follows:

$$P_{g-sky}(t) = P(t) \times \prod_{t' \in D_1, t' \prec t} (1 - P(t')) \times \prod_{t' \in D_2, t' \prec t} (1 - P(t')) \times \dots \times \prod_{t' \in D_m, t' \prec t} (1 - P(t')) \tag{4}$$

If we unify all the distributed databases D_i into one database D, that is $D = D_1 \cup D_2 \cup \dots \cup D_m$, then equation (4) can be rewritten as:

$$P_{g-sky}(t) = \prod_{t' \in D, t' \prec t} (1 - P(t')) \times P(t) \tag{5}$$

Note that, without loss of general situations as studied for many problems over distributed data, a critical requirement in distributed computation is to have low bandwidth consumption, thus, in this paper our main objective is to minimize the total communication cost in computing the global skyline result. The commonly used symbols are summarized in Table I.

TABLE I FREQUENTLY USED SYMBOLS

Symbol	Meaning
H	The central server
m	The number of local sites
S_i	The i-th local site
D_i	The i-th uncertain database
D	The global uncertain database
$P_{g\text{-}sky}(t)$	The global skyline probability of tuple t
$SKY(H)$	The set of global skyline tuples obtained by H
$SKY(D_i)$	The set of local skyline tuples from D_i

B. The Baseline Approach

A straightforward solution for our problem can always ask all local sites to transmit their uncertain databases to the server H, and then process the skyline query at H locally. Suppose we have a centralized uncertain database $D=\{ t_1, t_2, \ldots, t_N\}$ at server H, according to the calculation semantic in equation (3), a brute-force algorithm needs $O(N)$ time to compute the skyline probability of a tuple t_i, and $O(N^2)$ time to compute all tuple's skyline probabilities and select these qualified tuples.

However, this approach is both communication- and computation-expensive, the total communication cost is $|D|= \sum_{i=1}^{m} |D_i|$. As mentioned in [22], a good distributed skyline algorithm should achieve the following goals:

1. Minimization of bandwidth consumption. We measure bandwidth in the total number of tuples transmitted over the network. Precisely speaking, extra bandwidth is needed for sending synchronizing messages and the packet headers in order to enforce the used network protocol. Since these messages and headers are very small compared to transmitted tuples, they are ignored in our work.

2. Progressiveness. Ideally, a good algorithm should quickly return some early results soon after the beginning and produce a majority of the remaining results well before the end of execution.

The baseline approach is clearly very expensive and hard to achieve the above goals, thus we have an even more urgent need for effective and efficient processing techniques for skyline in the distributed uncertain data, which are the main contribution of this paper.

IV. The General Framework

Fig.3 illustrates the general framework for answering distributed skyline queries over uncertain data. In particular, at the beginning of DSUD, each participant S_i computes its local skyline using a centralized skyline algorithm (the detailed implementation will be introduced in Section VI) and sorts the skyline points in descending order of their skyline probabilities. At the same time, the centralized coordinator H initializes an empty set $SKY(H)$, where all the global skyline tuples will be contained eventually. Then, the rest of the algorithm is carried out in iterations, until all of local skyline sets $SKY(D_i)$ ($1 \le i \le m$) have become empty or the largest local skyline probability of any tuple within D_i is smaller than the user specified threshold q. Note that, each iteration consists of four phases:

1. To-Server Phase. In the initial phase, each local site S_i selects its representative tuple in $SKY(D_i)$ with the largest skyline probability to the coordinator H. After completing the first iteration, which site should be selected again to transfer its representative tuple depends on the results of the Server-Delivery phase.

2. Server-Calculation Phase. In this phase, H unions all the representative skylines from different sites into a small data set D_0, and retrieves the skyline $SKY(D_0)$ of D_0 again using a centralized skyline algorithm. Note that, after all the skyline points are selected from D_0, they are arranged in descending order of their partial skyline probabilities over D_0.

3. Server-Delivery Phase. Central server H selects a tuple from $SKY(D_0)$ with the largest partial skyline probability and broadcasts it to the other local sites (except the site where the tuple comes from) to obtain its global skyline probability. Note that, suppose that H broadcasts tuple t_{ij} belonging to site S_i, then the representative tuple will be selected in S_i in the next To-Server phase.

4. Local-Pruning Phase. When the candidate tuple from sever H is delivered to the local sites, a local pruning method is invoked to prune all those unqualified skyline tuples from $SKY(D_i)$. It can be verified that the pruned skyline is guaranteed not to be the global skyline of the global data set D.

```
Procedure DSUD_framework
Input : A number of k local uncertain databases D_i, a threshold q
Output : a set SKY(H) containing those qualified skyline tuples
BEGIN
(1) compute the local skyline SKY(D_i) over D_i; //local computing phase
(2) sort each SKY(D_i) in descending order of local skyline probability;
(3) while SKY(D_i) is not empty  // iteration
(4)    each site sends its representative tuple in SKY(D_i) to H;
(5)    compute each tuple's local skyline probability within H;
(6)    select representative tuples from H and deliver them to local sites;
(7)    obtain the global skyline probability, and add to SKY(H);
(8)    prune unqualified tuples in local site;
(9) end while;
END
```

Fig. 3 The framework for DSUD processing

In the following section, we will prove the correctness of restricting the solution to this framework and show the detail of DSUD algorithm, as well as its enhanced version e-DSUD.

V. The Algorithms

A. Detail of DSUD Algorithm

Although the baseline approach is correct, it violates all the performance targets as described above. An important improvement to this approach is as follows. First, each local database computes its local skyline and sends them to the central server H. Then, H unions all these skylines into another dataset, and broadcasts every tuples within this dataset to all the other local sites to obtain the global skyline probability. However, the union of all the local skylines may have a very large size, which still incurs expensive bandwidth consumption. In fact, it is easy to observe that not all the local skyline tuples are eventually in the final global skyline set, and thus would not need to be transferred to the server H. This observation indicates that an optimistic algorithm should be able to transfer these unqualified tuples as less as possible, which makes an important rationale behind the DSUD algorithm.

Another important observation is that the global skyline probability of any tuple t_{ij} could be calculated accumulatively from all the local sites. In particularly, we have the following Observation and Lemma.

Observation 1. For any tuple t_{ij} from uncertain database D_i at the local site S_i, the local skyline probability of t_{ij} to another uncertain database D_x at local site S_x ($x \neq i$) can be defined as follow:

$$P_{sky}(t_{ij}, D_x) = \prod_{t' \in D_x, t' \prec t_{ij}} (1 - P(t')) \tag{6}$$

Note that, since tuple t_{ij} is not belonging to database D_x, thus its local skyline probability in D_x is not affected by its existential probability $P(t_{ij})$.

Based on the above observation, we have the following lemma to calculate the global skyline probability of t_{ij} from database D_i.

Lemma 1. Given a number m of an uncertain databases D_i ($1 \leq i \leq m$) with size n_i, the global skyline probability of t_{ij} from D_i, denoted as $P_{g\text{-}sky}(t_{ij})$, equals to:

$$P_{g-sky}(t_{ij}) = \prod_{x=1}^{m} P_{sky}(t_{ij}, D_x) \tag{7}$$

Proof. The proof of this lemma is easy, according to equation (5), we have:

$$\begin{aligned} P_{g-sky}(t_{ij}) &= P(t_{ij}) \times \prod_{t' \in D_1, t' \prec t_{ij}} (1 - P(t')) \times \prod_{t' \in D_2, t' \prec t_{ij}} (1 - P(t')) \\ &\quad \times \ldots \times \prod_{t' \in D_m, t' \prec t_{ij}} (1 - P(t')) \\ &= P(t_{ij}) \times \prod_{t' \in D_i, t' \prec t_{ij}} (1 - P(t')) \times \prod_{s=1, s \neq i}^{m} \prod_{t' \in D_s, t' \prec t_{ij}} (1 - P(t')) \\ &= P_{sky}(t_{ij}, D_i) \times \prod_{s=1, s \neq i}^{m} P_{sky}(t_{ij}, D_s)) \text{ //byequation(3)and(6)} \\ &= \prod_{i=1}^{m} P_{sky}(t_{ij}, D_i) \end{aligned}$$

It is easy to observe that by sending only the most promising tuple t_{ij} from site S_i to the other local sites, the global skyline probability of t_{ij} can be calculated accumulatively. In order to find the most promising tuples from each locate site, our proposed DSUD algorithm sorts all the tuples at site S_i based on their local skyline probabilities $P_{sky}(t_{ij}, D_i)$. Without loss of generality, we assume $P_{sky}(t_{i1}, D_i) \geq P_{sky}(t_{i2}, D_i) \geq \ldots \geq P_{sky}(t_{in}, D_i)$ for any local site s_i. Thus, these m local sites S_i transmit their most promising tuples to the central server H in descending order of their local skyline probabilities. More precisely, a priority queue L of size m is maintained at the server H, and each site S_i sends its representative tuple with its id, existential probability and $P_{sky}(t_{ij}, D_i)$ that corresponds to the local skyline probability, i.e., a quaternion $<i, j, P(t_{ij}), P_{sky}(t_{ij}, D_i)>$ to L. The quaternion in the priority queue is sorted by the local skyline probability in descending order. Clearly, the priority queue L is initialized by retrieving the first tuple's id, existential probability and local skyline probability from each local site S_i.

In the next sever-calculation step, H retrieves the first element from priority queue L, i.e., $<i, j, P(t_{ij}), P_{sky}(t_{ij}, D_i)>$, which is considered as one of the most competing global skyline tuples. In order to compute the exact global skyline probability of t_{ij} that H has just obtained, H broadcasts t_{ij} to all the other local sites except S_i and asks each site D_x to report back the value $P_{sky}(t_{ij}, D_x)$ (based on equation (6)). By Lemma 1, H obtains the exact global skyline probability $P_{g\text{-}sky}(t_{ij})$ of tuple t_{ij} according to the returned values. Meanwhile, the feedback t_{ij} from server H will be used to prune these unqualified skyline tuples from $SKY(D_x)$. Then, after tuple t_{ij} is fetched from L, H asks for another tuple $t_{i(j+1)}$ from site S_i, along with its id, existential probability and the local skyline probability $P_{sky}(t_{ij}, D_i)$ which consists a quaternion $<i, j+1, P(t_{i(j+1)}), P_{sky}(t_{i(j+1)}, D_i)>$ to be inserted into the priority queue L. Up to here, one iteration is completed.

Based on Lemma 1, we have the following corollary which could be used to guarantee the iteration halt timely and correctly:

Corollary 1. The global skyline probability $P_{g\text{-}sky}(t_{ij})$ of a tuple is upper bounded by its local skyline probability $P_{sky}(t_{ij}, D_i)$, that is to say: $P_{g\text{-}sky}(t_{ij}) \leq P_{sky}(t_{ij}, D_i)$.

Note that, the local skyline probability of any unfetched tuples by H from all local sites is upper bounded by the head element from the priority queue L. Thus, it is easy to induce that, the global skyline probability of any unfetched tuples in all the local sites is upper bounded by the local skyline probability of the first element within L. Let $P_{sky}(t_{ij}, D_i)$ be the local skyline probability of the first element in L. It is safe for the iteration to terminate if $P_{sky}(t_{ij}, D_i) < q$ at a certain round. We denote this algorithm as DSUD.

In the following, we introduce an enhanced DSUD algorithm, which selects the feedback in an effective way and can make the iteration terminate early.

B. The Enhanced DSUD

Rationale. Given a set of local skyline tuples discovered at central server H so far, the best selection criterion is to select one tuple with the most powerful dominant ability as the feedback for every participant S_i. In particular, the criterion adopts the following principle: a tuple t in server H is currently the most promising feedback to local site S_i, only if it has the largest global skyline probability upper bound, that is to say, tuple t can prevent local sites from transmitting more tuples to server H. Note that, if no feedback was used, each local site S_i would need to transfer the entire local skyline set $SKY(D_i)$ to central server H. In order to select the most dominant tuple from priority queue L, H must obtain some knowledge about the global skyline probabilities of these tuples gathered in L or the content of local skyline set $SKY(S_i)$. Otherwise, H cannot figure out how many tuples would be pruned by a feedback tuple. A naïve approach for H to obtain such information would be to ask all the local sites S_i to transfer some sketch of the data distribution over uncertain database D_i. However, this approach itself is communication expensive for the transmitting of such sketch may occupy too much network bandwidth. Thus, a better solution is to calculate the approximation of the global skyline probability of the tuple within L without further bandwidth consumption. We have the following observation.

Observation 2. For tuple t from uncertain database D_i at local site S_i, and another tuple s from uncertain database D_j at site S_j. Suppose both of these two tuples are fetched into priority queue L, and tuple s is dominated by t, then the upper bound of the local skyline probability of s to uncertain database D_i at local site S_i can be defined as follows:

$$\begin{aligned} P_{sky}(s, D_i) &= \prod\nolimits_{t' \in D_i, t' \prec t} (1 - P(t')) \times (1 - P(t)) \\ &\quad \times \prod\nolimits_{s' \in D_i, s' \prec s, s' ! \prec t} (1 - P(s')) \\ &= \frac{P_{sky}(t, D_i)}{P(t)} \times (1 - P(t)) \times \prod\nolimits_{s' \in D_i, s' \prec s, s' ! \prec t} (1 - P(s')) \\ &\le \frac{P_{sky}(t, D_i)}{P(t)} \times (1 - P(t)) \end{aligned} \quad (8)$$

From the above observation, we can conclude that the upper bound of the local skyline probability of tuple s to another uncertain database D_i would be easily obtained by only using the information in priority queue L. Furthermore, the following corollary can be used to deduce a more precise upper bound of the global skyline probability of a tuple s within the priority queue L.

Corollary 2. The global skyline probability $P_{g\text{-}sky}(s)$ of a tuple s from site S_j is upper bounded by its local skyline probability $P_{sky}(s, D_j)$ multiplies the upper bound of its local skyline probabilities to other uncertain databases D_x, from where its representative tuple t transferred to priority queue L is in dominant of tuple s, that is to say:

$$\begin{aligned} P_{g-sky}(s) &= \prod_{x=1}^{m} P_{sky}(s, D_x) \\ &= P_{sky}(s, D_j) \times \prod_{x=1, x \ne j}^{m} P_{sky}(s, D_x) \\ &\le P_{sky}(s, D_j) \times \prod_{x=1, x \ne j, t \in D_x, t \in L, t \prec s}^{m} P_{sky}(s, D_x) \\ &\le P_{sky}(s, D_j) \times \prod_{x=1, x \ne j, t \in D_x, t \in L, t \prec s}^{m} \frac{P_{sky}(t, D_x)}{P(t)} \times (1 - P(t)) \\ &= P_{g-sky}(s)^* \end{aligned} \quad (9)$$

The proof of this corollary is obvious, we will not show the details here.

The approximate value of the global skyline probability $P_{g\text{-}sky}(s)$, denoted as $P_{g\text{-}sky}(s)^*$, can be obtained without more bandwidth consumption. Up to now, we are ready to elaborate how to choose optimal feedback from priority queue L to all the local sites. For this purpose, the coordinator H needs to compute $P_{g\text{-}sky}(s)^*$ of those tuples obtained in priority queue L so far and then choose the tuple with largest approximate value as the feedback. In particular, these tuples with its id, existential probability and the approximate value, i.e. a quaternion $<i, j, P(t_{ij}), P_{g\text{-}sky}(t_{ij})^*>$ will be inserted into another priority queue G sorted in descending order of their global skyline approximate values. Then, H selects the first element in the sorted queue and broadcasts it to other local sites to obtain the exact global skyline probability $P_{g\text{-}sky}(t_{ij})$. If $P_{g\text{-}sky}(t_{ij})$ is not smaller than threshold q, then t_{ij} will be added to the global skyline set $SKY(H)$. In the sequel, the next tuple from $SKY(D_i)$ at site S_i will be sent to coordinator H, and then the next iteration starts.

Note that, if the approximate value $P_{g\text{-}sky}(s)^*$ of a global skyline probability $P_{g\text{-}sky}(s)$ is smaller than threshold q, then tuple s will be expunged from the system immediately, and a tuple from the same local source in $SKY(D_i)$ will be retrieved to coordinator H in the next round of iteration.

C. An Illustrative Example

In this sub-section, we will illustrate our proposed e-DSUD algorithm with a concrete example. Suppose a hotel booking system that has three local sites: S_1 in Qingdao, S_2 in Shanghai, and S_3 in Xiamen. In particular, two dimensional attributes, price of the hotel and distance to the beach, of a tuple are contained in each local database at the local site. Besides, each tuple is also associated with a confidence (existential) probability. For example, a triple <340, 66, 0.8> means the price of a hotel is 340, its distance to the beach is 66, and its confidence probability is 0.8. Now, a customer wants to query the skyline set over these three cities, and quality guarantee threshold q is set to 0.3. The query will be processed as follows.

At the beginning of e-DSUD, each local site computes its own local skyline set, and put all these qualified skyline tuples in $SKY(D_i)$. Table II(a) illustrates these skyline tuples at each site D_i ($1 \le i \le 3$), sorted in descending order of their skyline probabilities. More precisely, the quaternion <3, 8, 0.8, 0.5> means the existential probability of tuple <3, 8> is 0.8, and its local skyline probability within uncertain database D_1 is 0.5.

In the To-Server phase, central server H is initialized with each local site's representative tuple at the first iteration, that is, each participant S_i sends its first tuple in $SKY(D_i)$ to server H. In particular, H retrieves (6, 6, 0.7, 0.65) from $SKY(D_1)$, (6.5, 7, 0.8, 0.65) from $SKY(D_2)$, and (6.4, 7.5, 0.9, 0.8) from $SKY(D_3)$. These items are stored according to their local skyline probabilities in a priority queue L at server H, shown in Table II(b).

In order to select the most dominant tuple from L, in the server-computation phase, we use equation (9) to calculate the approximate values of the global skyline probabilities for these tuples in L. In particular, since tuple (6.4, 7.5) is dominated by tuple (6, 6), its approximate value equals to $P_{g\text{-}sky}((6.4, 7.5))^* = P_{sky}((6.4, 7.5), D_3) \times P_{sky}((6, 6), D_1)/P(6,6) \times (1\text{-}P(6, 6)) = 0.8 \times (0.65/0.7) \times 0.3 = 0.22$, similarly, $P_{g\text{-}sky}((6.5, 7))^* = P_{sky}((6.5, 7), D_2) \times P_{sky}((6, 6), D_1)/P(6,6) \times (1\text{-}P(6, 6)) = 0.18$. Note that, tuple (6, 6) is a skyline tuple within L, thus its approximate probability $P_{g\text{-}sky}((6, 6))^*$ remains the same as $P_{sky}((6, 6), D_1) = 0.65$. Consequently, in the following feedback phase, coordinator H selects tuple (6, 6) which has the largest skyline value and sends it to the other two local sites D_2 and D_3, where the local skyline probability of tuple (6, 6) is obtained to get its final global skyline probability (equation

(7)). Suppose the global skyline probability $P_{g\text{-}sky}((6, 6))$ is larger than threshold 0.3, then (6, 6) will be added to *SKY*(*H*). Meanwhile, in the local-pruning phase, D_2 removes (6.5, 7, 0.8, 0.65) from *SKY*(D_2), and D_3 removes (6.4, 7.5, 0.9, 0.8) from *SKY*(D_3), since they are dominated by (6, 6). At the end of the first iteration, the updated list of active local skyline tuples at each local site is shown in Table II(c).

In the sequel, the second iteration starts. Different from the first iteration, only local site D_1 transmits tuple (8, 4, 0.8, 0.6) in its current local skyline list *SKY*(D_1) to *H*. Table II(d) demonstrates the tuples gathered by server *H* so far. The server computation phase discovers a new skyline tuple (8, 4, 0.8, 0.6), which has the largest approximate skyline value and is returned as the feedback to other two local sites D_2 and D_3. For the pruning power of feedback information, tuple (8, 4) will filter (9, 5, 0.7, 0.6) out of *SKY*(D_2), and (10, 4.5, 0.7, 0.7) out of *SKY*(D_3). Table II(e) presents the current active local skyline tuples within each local site. Again, suppose $P_{g\text{-}sky}((8, 4))$ is larger than threshold 0.3, then (8, 4) will be added to *SKY*(*H*). Thus, *SKY*(*H*) contains (6, 6) and (8, 4) up to now.

Similar to the second iteration, To-Server phase in the third iteration will transmit the one and only tuple (3, 8, 0.8, 0.5) in S_1's current local skyline list *SKY*(D_1) to *H*, as shown in Table II(f). Then, server *H* selects tuple (3, 8) and broadcasts it to local sites D_2 and D_3 to obtain its global skyline probability. Accordingly, in the local-pruning phase, D_2 removes (4, 9, 0.6, 0.6) from *SKY*(D_2), D_3 removes (3.5, 11, 0.7, 0.7) from *SKY*(D_3), since they are dominated by (3, 8). Suppose the global skyline probability $P_{g\text{-}sky}((3, 8))$ is larger than 0.3, then (8, 4) will be added to *SKY*(*H*). Thus, *SKY*(*H*) is updated to (6, 6), (8, 4), and (3, 8). Up to now, since all the local skyline set *SKY*(D_i) ($1 \le i \le 3$) is empty as described in Table II(g), and the largest skyline probability for tuples within central server is smaller than 0.3 as shown in Table II(h), the exit condition holds and our e-DSUD algorithm terminates.

TABLE II ILLUSTRATION OF E-DSUD ALGORITHM

Set	Content
SKY(D_1)	(6, 6, 0.7, 0.65), (8, 4, 0.8, 0.6), (3, 8, 0.8, 0.5)
SKY(D_2)	(6.5, 7, 0.8, 0.65), (4, 9, 0.6, 0.6), (9, 5, 0.7, 0.6)
SKY(D_3)	(6.4, 7.5, 0.9, 0.8), (3.5, 11, 0.7, 0.7), (10, 4.5, 0.7, 0.7)

(a)

Server	Content
L	(6.4, 7.5, 0.9, 0.8), **(6, 6, 0.7, 0.65)**, (6.5, 7, 0.8, 0.65)

(b)

Set	Content
SKY(D_1)	(8, 4, 0.8, 0.6), (3, 8, 0.8, 0.5)
SKY(D_2)	(4, 9, 0.6, 0.6), (9, 5, 0.7, 0.6)
SKY(D_3)	(3.5, 11, 0.7, 0.7), (10, 4.5, 0.7, 0.7)

(c)

Server	Content
L	**(8, 4, 0.8, 0.6)** , (6.4, 7.5, 0.9, 0.22), (6.5, 7, 0.8, 0.18)

(d)

Set	Content
SKY(D_1)	(3, 8, 0.8, 0.5)
SKY(D_2)	(4, 9, 0.6, 0.6)
SKY(D_3)	(3.5, 11, 0.7, 0.7)

(e)

Server	Content
L	**(3, 8, 0.8, 0.5)**, (6.4, 7.5, 0.9, 0.24), (6.5, 7, 0.8, 0.195)

(f)

Set	Content
SKY(D_1)	NULL
SKY(D_2)	NULL
SKY(D_3)	NULL

(g)

Server	Content
L	(6.4, 7.5, 0.9, 0.24), (6.5, 7, 0.8, 0.195)

(h)

VI. THE IMPLEMENTATION ISSUES

Two trivial executions of our proposed algorithm are: 1) Calculate the local skyline set *SKY*(D_i) in each local site S_i and then send the first element to coordinator *H*; 2) Broadcast each element in the priority queue to other local sets to obtain its global skyline probability $P_{g\text{-}sky}(a)$; then choose element with $P_{g\text{-}sky}(a) \ge q$ to the final skyline set *SKY*(*H*). Note that, the basic linear scan method may cause all the elements in database to be attached for computing the skyline probability. Thus, the amortized time complexity is $O(|N|)$ per element which is too pessimistic for query processing.

For managing a large number of elements, indexes are usually used to improve query efficiency. Currently, the most popular indexing technique for multidimensional data storage and access is the well known R-tree [10]. In this section, we present novel techniques to efficiently execute algorithms based on R-tree with the aim to visit as few elements as possible. Based on these techniques, we incrementally maintain the priority queue *L* and global skyline set *SKY*(*H*).

A. The Probabilistic R-tree

This section explains the structure of a *Probabilistic R-tree* (PR-tree) and its properties. A PR-tree is designed for pruning sub-trees that do not contain any qualified skyline results, thus to accelerate the skyline query procedure. The general idea of PR-tree is to allow the entry of node in R-tree to store probability values. That is, each leaf entry contains not only the tuple's attributes, but also the existential probability of the tuple.

In particular, at each intermediate entry *e* in PR-tree, the following information is stored: a pointer referencing its child node, a minimum bounding rectangle which includes all the objects in the child node of *e*, an existential probability P_1 which equals to the smallest occurrence probability of the elements rooted at *e*, and an existential probability P_2 which equals to the largest occurrence probability of the elements rooted at *e*.

B. Local Skyline Query Procedure

Given a user specified probability threshold *q*, this section explains how to obtain the qualified local skyline set efficiently through the usage of PR-tree. Our proposed query algorithm is similar to the I/O optimal *Branch and Bounding*

Skyline (BBS) algorithm proposed in [11]. In particular, the local skyline query procedure starts at the root node of the PR-tree and inserts all its entries into a heap sorted according to their minimum distance *mindist* to the space origin. The entry with the minimum *mindist* is selected to expand. This expansion removes the selected entry out of the heap firstly, and then inserts its children into the heap. The procedure continues and the next expanded entry is again the one with minimum *mindist*. When the element popped from the heap is any object a in the leaf node, and suppose the occurrence probability of a is larger than q, then object a belongs to the local skyline, and is inserted to the local skyline list $SKY(D_i)$. Otherwise, a is abandoned by the algorithm. The algorithm proceeds in the same manner until the heap becomes empty.

Note that, before an entry is inserted into the heap or is expanded, it has to be guaranteed that it contains or itself is a qualified local skyline. More precisely, suppose an intermediate entry b is dominated by an object a contained in local database D_i, and the largest occurrence probability P_2 of b multiplies the non-occurrence probability of object a is smaller than threshold q, then there is no need to insert entry b into the heap, as there is no qualified skylines contained in the sub-tree of b.

C. Global Skyline Query Procedure

As explained above, to get the global skyline probability of a tuple t, the critical issue is to get its local skyline probabilities over all the local databases. We use the PR-tree to speed up the calculation of local skyline probabilities. In particular, for uncertain tuple b that broadcasted from the central server H. Suppose we calculate the local skyline probability of b against local database D_i, then we issue a window query (with space origin and the location of b as diagonal corners) on the PR-tree constructed over D_i to obtain those tuples dominate b. From these uncertain tuples returned by this query, we calculate the local skyline probability of b over D_i by multiplies their non-occurrence probabilities.

VII. EXPERIMENTAL RESULTS

In this section, we empirically evaluate the effectiveness and efficiency of the proposed DSUD algorithm and its enhanced version e-USDU algorithm. Since the baseline approach, which simply asks all the participants to send their entire uncertain databases to the coordinator, is too communication expensive. We will consider the DSUD algorithm as the baseline algorithm, and compare e-DSUD against this baseline solution.

Data set. We use both the synthetic and real data sets for experiments. In particular, we create synthetic data with two popular distributions in the literature [5]: Independent and Anti-correlated, whose generation follows the description in Fig.4. The dimensionality d of a synthetic data set is a parameter ranging from 2 to 5. The overall cardinality of the generated dataset is 2 million tuples. Furthermore, we use uniform distribution to randomly assign an occurrence probability of each tuple to make them be uncertain. Clearly, in uniform distribution, the occurrence probability p of each tuple takes a random value between 0 and 1.

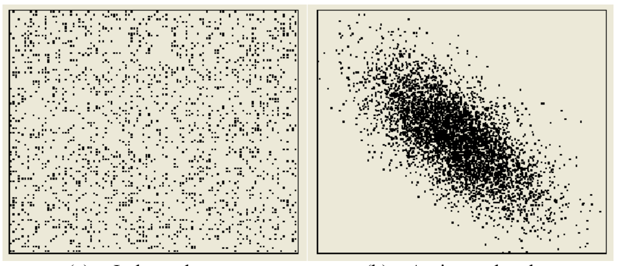

Fig. 4. Illustration of synthetic data

After the synthetic dataset is generated, and given the number of m local sites, each tuple from the synthetic uncertain database D is assigned to site S_i chosen uniformly. Clearly, all local sites have the same data distribution. In particular, a local site server keeps a random sample set of the underlying data set, and the sample sets are mutually disjoint. In the experiments, every local server possesses an equal number of points, named the local cardinality. In case of m local sites, the local cardinality $|D_i|$ equals to $|N|$/m, where $|N|$ is the cardinality of the entire data set.

User specified q is the probability threshold in evaluating the efficiency of query processing. To evaluate both the DSUD and e-DSUD algorithms, we use 0.3 as a default value of q, and evaluate its effect with 3 given probability thresholds 0.5, 0.7, and 0.9 that spread between [0.3, 1].

In the following experiments, we evaluate the efficiency of our algorithms, in terms of bandwidth consumption against dimensionality d, number of local databases m, and probabilistic threshold q under two distributions of objects' spatial locations. Specifically, bandwidth consumption is measured by the number of tuples transmitted over the network. Table III summarizes the parameters, as well as their values, to be examined in our experiments. Unless specifically pointed out, each parameter is set to its default value as indicated in bold in Table III.

TABLE III SYSTEM PARAMETERS

Parameter	Values
N	2000K
m	40 **60** 80 100
d	2 **3** 4 5
q	**0.3** 0.5 0.7 0.9
P	[0, 1]

A. Performance with Dimensionality

Figs. 5(a) and 5(b) illustrate the total bandwidth consumption as a function of dimensionality d, under the Independent and Anti-correlated distributions, respectively. The numbers of transmitted tuples over two algorithms increase when d varies from 2 to 5. This is as expected, since larger the dimensionality of data tuples would make more tuples not to be dominated with each other, which makes the final skyline set become larger. Obviously, our e-DSUD algorithm requires considerably less bandwidth than its baseline DSUD algorithm, which indicates the efficiency and great importance of our proposed feedback mechanism in e-DSUD algorithm. Furthermore, we also observe that the anti-correlated data sets always have the larger bandwidth consumption than the independent data distribution under the same experimental settings. This is similar to the situations of skylines on centralized database as the dimensionality grows.

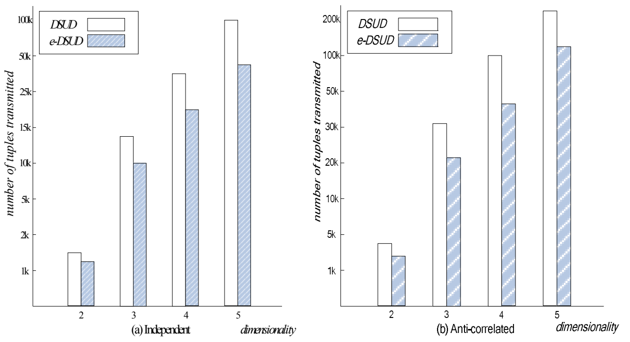

Fig. 5. Performance vs. Dimensionality d

B. Performance with Number of Local Sites

Figs. 6(a) and 6(b) show the total bandwidth consumption when we vary m from 40 to 100, under the Independent and Anti-correlated distributions. The numbers of transmitted tuples by two algorithms increase when m gets larger. This is reasonable, in the feedback phase, a tuple selected from central server H is broadcasted to all the local sites S_i. Thus, m local sites mean at least a number of m tuples are transmitted over the network within one iteration. Since the total number of final skyline tuples, which should be delivered from the H, is fixed according to the generated database, thus larger the number of local sites would make more network bandwidth consumption. Also, our proposed e-DSUD algorithm requires less bandwidth than baseline DSUD algorithm, which indicates the pruning efficiency of our proposed feedback mechanism in e-DSUD processing.

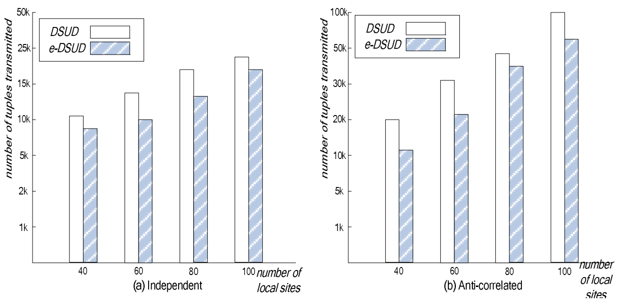

Fig. 6. Performance vs. Number of local sites m

C. Performance with Threshold q

As shown in both Figs. 7(a) and 7(b), we find that, the numbers of transmitted tuples over two algorithms decrease when q gets larger. This is as expected, since the probability threshold p of the skyline query affects the total size of the final qualified skylines. Generally speaking, the smaller the probability threshold, the larger the skyline results. The reason is that if a tuple M belongs to p-Skyline, then it will always be in the result of p'-Skyline, where $p' \leq p$. Thus, since the total number of final skyline tuples, which should be delivered over the network, decreases according to the increasing threshold q, the larger value of q would make less network bandwidth consumption. Note that, the query performance is very sensitive to the change of probability threshold. This is because the feedback mechanism used in our framework can prune most of the unqualified skyline candidates under larger threshold.

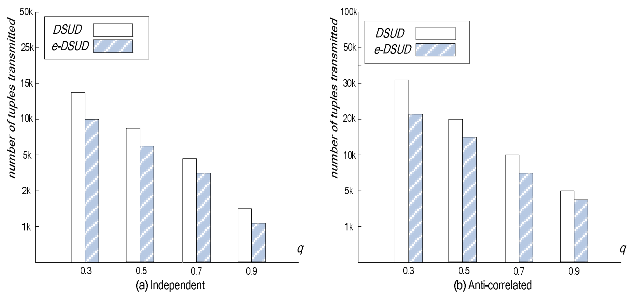

Fig. 7. Performance vs. Threshold q

D. Performance with Real Dataset

In the last set of experiments, we use the real dataset NYSE borrowed from [2] to evaluate the query performance of our proposed algorithms. In particular, NYSE is extracted from the stock statistics from New York Stock Exchange, and it records 2 million stock transaction of Dell Incorporation from 1/12/2000 to 22/5/2001. Each transaction (tuple) has two attributes: the average price per volume and total number of volume. Similarly, in order to represent the transaction (tuple) uncertainty, we randomly assign a probability value to each transaction, following uniform distribution. The entire NYSE dataset is assigned to m local sites equally.

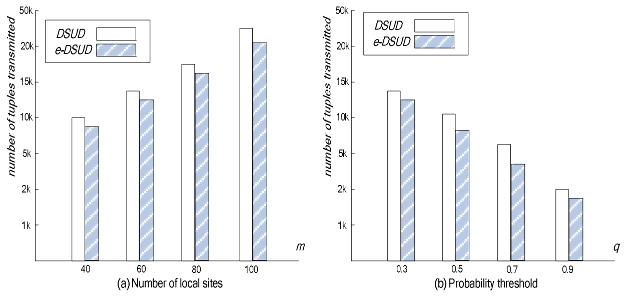

Fig. 8. Performance vs. Real dataset

As shown in both Figs. 8(a) and 8(b), the bandwidth consumptions of our proposed algorithms over NYSE are reported by varying number of local sites m from 40 to 100 (the default probability threshold is set to 0.3), and the probability threshold q from 0.3 to 0.9 (the default number of local sites is set to 60). In Fig.8(a), the numbers of transmitted tuples over two algorithms increase when m gets larger. This trend is similar to the synthetic dataset in Fig. 6. Similarly, the numbers of transmitted tuples over two algorithms drop as threshold hold q increases.

VIII. Conclusions

This is the first work that studies skyline queries for distributed probabilistic data. We show that significant communication cost could be saved by exploring the interplay between the global skyline probability and its approximate value. Through the usage of PR-tree, we also demonstrate how to alleviate the computation burden at each distributed site so that communication and computation efficiency are achieved simultaneously. Experiments have been conducted to verify that our techniques can process skyline queries over distributed uncertain data both communication- and computation-effectively.

In this paper, we have focused on horizontal partitioning, where a local site has all the attributes but stores only a subset of the entire tuples. As an interesting direction, vertical partitioning between distributed data still exists in the context of uncertain data. Thus, studying new algorithms to those cases is an important future work. Finally, when updates occur at the distributed sites, how to incrementally maintain the global skyline results is also an interesting problem.

IX. Acknowledgment

The work is supported by the Key Project in the National Science and Technology Pillar program of China under grant No.2008BAH29B00 and Central Colleges of basic scientific research and operational costs under grant No.Q2009019.

References

[1] F. Li, K. Yi, and J. Jestes, "Ranking Distributed Probabilistic Data", In *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD'09)*, Providence, RI, June 2009.

[2] W. Zhang, X. Lin, Y. Zhang, W. Wang, and Jeffrey Yu, "Probabilistic Skyline Operator over Sliding Windows", In *Proc. International Conference on Data Engineering (ICDE'09)*, pp.305-316, Shanghai, China, March 2009.

[3] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, "Efficient Computation of the Skyline Cube", In *Proc. 31st International Conference on Very Large Data Bases (VLDB'05)*, pp.241-252, 2005.

[4] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting Stars: The K Most Representative Skyline Operator", In *Proc. 23rd International Conference on Data Engineering (ICDE'07)*, pp.86-95, 2007.

[5] S. Borzsonyi, D. Kossmann, and K. Stocker, "The Skyline Operator", *Proc. 17th International Conference on Data Engineering* (ICDE'01), pp.421-430, 2001.

[6] X. Lian and L. Chen, "Monochromatic and Bichromatic Reverse Skyline Search over Uncertain Database", In *Proc. ACM SIGMOD International Conference on Management of Data*, pp.213-226, 2008.

[7] X. Lian and L. Chen, "Probabilistic Ranked Queries in Uncertain Databases", In *Proc. International Conference on Extending Database Technology (EDBT'08)*, pp.511-522, 2008.

[8] M. Hua, J. Pei, W. Zhang, and X. Lin, "Ranking queries on uncertain data: A probabilistic threshold approach", In *Proc. ACM SIGMOD International Conference on Management of Data*, 2008.

[9] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. M. Jermaine, and P. J. Haas, "MCDB: a monte carlo approach to managing uncertain data", In *Proc. ACM SIGMOD International Conference on Management of Data*, 2008.

[10] A. Guttman, "R-trees: a dynamic index structure for spatial searching", In *Proc. ACM SIGMOD International Conference on Management of Data*, pp.47-57, 1984.

[11] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries", In *Proc. ACM SIGMOD International Conference on Management of Data*, pp.467-478, 2003.

[12] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi, "Holistic aggregates in a networked world: distributed tracking of approximate quantiles", In *Proc. ACM SIGMOD International Conference on Management of Data*, 2005.

[13] R. Huebsch, M. Garofalakis, J. M. Hellerstein, and I. Stoica, "Sharing aggregate computation for distributed queries", In *Proc. ACM SIGMOD International Conference on Management of Data*, 2007.

[14] S. Wang, Q. H. Vu, B. C. Ooi, Anthony K. H. Tung, and L. Xu, "Skyframe: a framework for skyline query processing in peer-to-peer systems", *The VLDB Journa*l (2009) 18:345-362.

[15] S. Michel, P. Triantafillou, and G. Weikum, "KLEE: A framework for distributed top-k query algorithms", In *Proc. International Conference on Very Large Data Bases* (*VLDB*), 2005.

[16] I. Sharfman, A. Schuster, and D. Keren, "A geometric approach to monitoring threshold functions over distributed data streams", In *Proc. ACM SIGMOD International Conference on Management of Data*, 2006.

[17] B. Babcock and C. Olston, "Distributed top-k monitoring", In *Proc. ACM SIGMOD International Conference on Management of Data*, 2003.

[18] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks", In *Proc. International Conference on Very Large Data Bases* (*VLDB*), 2004.

[19] N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases", *The VLDB Journal*, 16(4):523–544, 2007.

[20] K. Deng, X. Zhou, and H. T. Shen, "Multi-source skyline query processing in road networks", In *Proc. 23rd International Conference on Data Engineering (ICDE'07)*, 2007.

[21] A. Vlachou, C. Doulkeridis, and Y. Kotidis, "Angle-based Space Partitioning for Efficient Parallel Skyline Computation", In *Proc. ACM SIGMOD International Conference on Management of Data* , 2008.

[22] L. Zhu, Y. Tao, and S. Zhou, "Distributed Skyline Retrieval with Low Bandwidth Consumption", In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2009: 21(3), 384-400.

[23] W.-T. Balke, U. Guntzer, and J. X. Zheng, "Efficient Distributed Skylining for Web Information Systems," In *Proc. Ninth Int'l Conf. Extending Database Technology (EDBT'04)*, pp.256-273, 2004.

[24] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis, "Skypeer: Efficient Subspace Skyline Computation over Distributed Data," In *Proc. 23rd Int'l Conf. on Data Engineering (ICDE'07)*, pp.416-425.

[25] P. Wu, C. Zhang, and Y. Feng, "Parallelizing Skyline Queries for Scalable Distribution," In *Proc. Int'l Conf. Extending Database Technology (EDBT'05)*, pp.112-130, 2005.

[26] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data", In *Proc. International Conference on Very Large Data Bases* (*VLDB*), 2007.

[27] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalable Content-Addressable Network," In *Proc. ACM SIGCOMM International Conference on Data Communications*, 2001.

[28] H. V. Jagadish, B. C. Ooi, and Q. H. Vu, "Baton: A Balanced Tree Structure for Peer-to-Peer Networks", In *Proc. International Conference on Very Large Data Bases* (*VLDB*), pp.661-672, 2005.

[29] A. Vlachou, C. Doulkeridis, K. Norvag, and M. Vazirgiannis, "On Efficient Top-k Query Processing in Highly Distributed Environments", In *Proc. ACM SIGMOD International Conference on Management of Data*, 2008.

[30] W.-T. Balke, W. Nejdl, W. Siberski, and U. Thaden, "Progressive distributed top-k retrieval in peer-to-peer networks," In *Proc. of IEEE Int. Conf. on Data Engineering (ICDE)*, pp.174-185, 2005.