Contents lists available at ScienceDirect
Big Data Research
www.elsevier.com/locate/bdr

GDPS: An Efficient Approach for Skyline Queries over Distributed Uncertain Data [☆]

Xiaoyong Li, Yijie Wang [*], Xiaoling Li, Xiaowei Wang, Jie Yu

National Key Laboratory for Parallel and Distributed Processing, College of Computer Science, National University of Defense Technology, Changsha, Hunan 410073, China

A R T I C L E I N F O

Article history:
Available online xxxx

Keywords:
Uncertain data
Probabilistic skyline
Distributed query
Grid summary

A B S T R A C T

The skyline query as an important aspect of big data management, 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. Moreover, the uncertain data from many applications have become increasing distributed, which makes the central assembly of data at one location for storage and query infeasible and inefficient. The lack of global knowledge and the computational complexity derived from the introduction of the data uncertainty make the skyline query over distributed uncertain data extremely challenging. Although many efforts have addressed the skyline query problem over various distributed scenarios, existing studies still lack the approaches to efficiently process the query. In this paper, we extensively study the distributed probabilistic skyline query problem and propose an efficient approach GDPS to address the problem with an optimized iterative feedback mechanism based on the grid summary. Furthermore, many strategies for further optimizing the query are also proposed, including the optimization strategies for the local pruning, tuple selecting and the server pruning. Extensive experiments on real and synthetic data sets have been conducted to verify the effectiveness and efficiency of our approach by comparing with the state-of-the-art approaches.

1. Introduction

In recent years, uncertain data management has received increasing attention with the emergence of many practical applications in domains like sensor network [1], RFID network [2], data cleaning and extraction [3], location-based service [4], market surveillance and social data collections [5]. The uncertainty is inherent in these applications, which may be derived from the noisy measurement, inference models, improper operator and consideration for privacy-preserving [6]. Due to the rapid increasing amount of data accumulated, analyzing large collections of uncertain data has become a challenging task.

Skyline operator as an important advanced query type, is necessary in order to help users to handle the huge amount of available data by identifying a set of interesting data objects. The skyline query is also known as the *Pareto-optimum* problem, which is a typical multi-objective optimization problem in nature. Given a set of multidimensional objects, the skyline query retrieves the objects in the set that are not *dominated* by others, where an object p_1 is said to dominate another object p_2, if p_1 is not worse than p_2 in all dimensions, but is strictly better than p_2 in at least one dimension. The most classical example for the skyline query is the hotels selection [7]. As shown in Fig. 1, each point in the 2-dimensional space (x, y) corresponds to a hotel record, where x and y axes represent the price of the hotel and the distance to the bench, respectively. The users usually only need to consider the points in the plotted line, as all the other points are dominated by the points in the line which are defined as skylines.

Recently, lots of efforts have been conducted to address the skyline queries over uncertain data [8–14]. However, all these studies focus on the query processing over centralized data sets, and cannot deal with the queries over distributed uncertain data. In reality, a large amount of uncertain data is collected by many emerging applications which contain multiple sources in a distributed manner, due to the increasing number of available data sources and the available network services. The typical scenarios include distributed sensor networks [15], distributed clouds [16] and multi-source data integration [17]. In these applications, data are usually collected from vast number of data sources among geographically scattered sites, which makes the central assembly of

☆ This article belongs to Scalable Computing for Big Data.
* Corresponding author.
E-mail addresses: sayingxmu@nudt.edu.cn (X. Li), wangyijie@nudt.edu.cn (Y. Wang), lixiaoling@nudt.edu.cn (X. Li), wangxiaowei@nudt.edu.com (X. Wang), yujie@nudt.edu.cn (J. Yu).

http://dx.doi.org/10.1016/j.bdr.2014.07.003
2214-5796/© 2014 Published by Elsevier Inc.

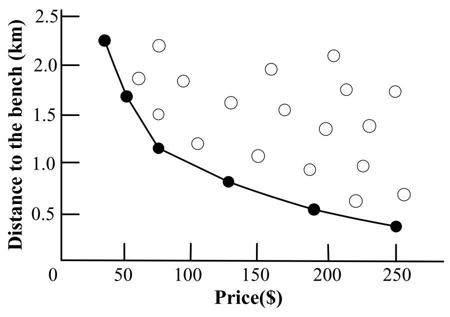


Fig. 1. An example of the skyline query.

data for storage and query at one location infeasible and inefficient [18]. Thus, until recently, there appear some studies target at distributed queries over uncertain data, such as the ranking queries [19,20] and join [21].

Actually, the distributed skyline query as an important aspect of big data management has many important applications. For instance, consider the services selection in distributed clouds where customers may want to select good services to fulfill their Quality of Service (QoS) requirements. The geographically located cloud providers offers large number of similar services at different prices and performance levels with different features, such as *response time*, *stability*, *accuracy*, *reliability*, *cost*, and *elasticity* [22]. Besides, the features are usually uncertain, due to the derivation from the user experiences or measured with software and hardware monitoring tools [22]. Thus, modeling the services as uncertain records and evaluating them with skyline queries to make recommendation for users has great significance.

Moreover, in the distributed stock trade systems, a trader needs to know which stocks worldwide are worth investing based on the historical trading records, which may be geographically distributed over a large number of sites. Generally, we can evaluate each stock based on multiple attributes like *average price*, *change*, *last close price*, *estimated price*, *volume*, etc. Furthermore, the recording errors caused by systems or improper operators by human beings may make failed deals be recorded successful, and vise versa. Thus, each stock can be viewed as an uncertain record. Accordingly, a skyline query against those distributed uncertain databases will help traders get those most interesting (skyline) stocks.

Another real-life example is online comparative shopping, in which a search engine needs to get good bargains from many distributed shopping sites according to multiple criteria like *price*, *quality*, *time*, etc. Moreover, we can additionally attach an attribute *credit* to represent the credibility of the record, which is usually determined by the feedbacks of the customers. Therefore, to evaluate the bargains fairly, we can model the evaluation as a distributed skyline query problem, aiming at retrieving the global skyline bargains over all distributed sites for decision-making. Besides all the above mentioned applications, the distributed uncertain skyline queries are also widely in many other domains like financial computing services or social networking services, such as the similarity match in Facebook and Twitter, as the data with uncertainty in the systems are geographically distributed over multiple data centers.

Skyline queries over certain data has received considerable attention in various distributed scenarios like P2P systems [13,23–27], web information systems [28], distributed data streams [29], and wireless sensor networks [30,31]. Nevertheless, the introduction of the data uncertainty makes the approaches cannot be applied to uncertain data. To the best of our knowledge, only the works [32,33] have investigated the distributed skyline query problem. Ding et al. [32] for the first time propose algorithms to address the query with an iterative feedback mechanism. However, the pruning capability of the algorithms is limited, as they adopted the local information to prune the unqualified tuples. If we can use some global knowledge of all the tuples, then the query performance can be further improved. Thus, our team publishes a short work and proposes a novel strategy to improve the queries based on the grid summary in [33]. Although our earlier proposed approach GBPS in [33] can improve the query efficiency, many problems for optimizations remain to be solved.

In this paper, we extensively investigate the distributed probabilistic skyline query problem and propose an efficient approach called *Grid summary based Distributed Probabilistic Skyline* (GDPS) to address the problem. The approach can achieve excellent progressiveness in outputting the results with much less bandwidth consumption. In summary, we make the following contributions in this paper:

- We formalize the distributed probabilistic skyline query and propose to use grid summary to capture the distribution and the global knowledge of the skyline probabilities for all the tuples, which can be used to the query optimization.
- We propose a general framework to process the skyline queries over distributed uncertain data with grid summary.
- We propose several novel strategies for the query optimization, which significantly further reduces the communication overhead.
- We conduct extensive experiments on real and synthetic data sets with real deployment under parameter settings to confirm the effectiveness and efficiency of our approach.

This paper significantly extends an earlier published conference paper [33] in many substantial ways. First, we provide more details about the background and the related work. Second, we describe the pruning process of the query with grid summary in more detail. Third, we introduce many strategies for improving the query with optimized iteration process. Finally, we reconsider the experimental settings and conduct more extensive experiments to evaluate the performance of the proposals. Through the experiments, we can find that the proposed approach GDPS is more efficient than the GBPS approach in [33].

The remainder of the paper is organized as follows. Section 2 provides a brief review of related work. In Section 3, we provide some preliminaries. Section 4 presents the grid summary, which can be seamlessly integrated to the following query optimization. Section 5 describes the general framework of GDPS approach for distributed probabilistic skyline computation in detail. In Section 6, we propose several novel strategies to further optimize the queries. The experimental evaluation is presented in Section 7. Finally, we conclude our paper in Section 8.

2. Related work

Skyline computation has recently attracted considerable attention in database community and experienced the overall trends from centralized queries to distributed or parallel queries, static skylines to dynamic (or relative) skylines, all space queries to subspace queries, certain data to uncertain data. Particularly, the distributed skyline queries and uncertain skyline queries are both important and still evolving research areas in database community [34,35].

Moreover, we observe that the two areas have been each studied quite extensively but separately, despite the fact that the two often arise concurrently in many applications. Therefore, we present the related work from two aspects: distributed skyline queries over certain data and uncertain skyline queries. Section 2.1 first reviews previous distributed skyline query processing techniques over certain data. Then, the related work of uncertain skyline queries is introduced in Section 2.2.

Table 1
Summary of approaches for distributed skyline queries.

Approaches	Network structure	Skyline query type	Partitioning	Environment
MANETs [30]	no overlay	global	data	P2P
QTree [36]	unstructured P2P	approximate	data	P2P
DSL [37]	CAN	constrained	space	P2P
SKYPEER [24]	super-peer	subspace	data	P2P
SWSMA [31]	no overlay	global	data	sensor network
SKYPEER+ [38]	super-peer	subspace	data	P2P
BITPEER [39]	super-peer	subspace	data	P2P
PaDSkyline [25]	no overlay	constrained	data	P2P
iSky [26]	BATON	global	space	P2P
SkyFrame [40]	BATON/CAN	global/approximate	space	P2P
FDS [27]	no overlay	global	data	P2P
AGiDS [41]	no overlay	global	data	P2P
BOCS [42]	no overlay	global	data	data streams
SkyPlan [42]	no overlay	global	data	P2P

2.1. Distributed skyline queries

The first distributed algorithm is proposed in [28], which focuses on the skyline query processing over vertical data distribution. Later, a number of algorithms for processing the distributed skylines over horizontal data distribution have been proposed in various scenarios, especially for the P2P environments. A comparative overview of distributed skyline literature is presented in Table 1.

Generally, the approaches that assume horizontal data distribution can be classified in two categories: the approaches with space partitioning and the approaches with data partitioning [41].

- *Approaches with space partitioning*: the approaches usually assume space partitioning among peers and each peer is responsible for a disjoint partition of the data space. Thus, the location of each data is controlled by the system and the peers are often organized as a particular structure to optimize the queries, such as structured P2P.
- *Approaches with data partitioning*: data partitioning is assumed and each peer autonomously stores its own data. Thus, the queries usually assume no particular network topology and no data reorganization among the peers in the system.

2.1.1. Approaches with space partitioning

To improve the queries, a structured P2P or special network overlay is often employed in this category of approaches. Specifically, DSL is proposed in [37] to process the constrained skyline query determined by CAN [43]. Wang et al. [40] propose *SkyFrame* based on a tree-based overlay (BATON) [44] for assigning data to peers. In addition, iSky is proposed in [26] to process the skyline also based on BATON, which employs another transformation iMinMax to assign data to the peers. The common of these approaches is that a server cannot freely decide the tuples in its own storage, whereas our techniques allow arbitrary horizontal partitioning.

2.1.2. Approaches with data partitioning

Since the data partitioning in this category is assumed, optimizing the queries by reassigning data is infeasible, thus many novel data structures are proposed to improve the query in many literatures. For example, Hose et al. [36] use distributed data summaries (*QTree*) to improve the routing to find the skyline points. Cui et al. [25] propose PaDSkyline by using of MBRs (Minimum Bounding Regions) to summarize the data stored at each server, and compute the skylines within each group using specific plans. This is improved by SkyPlan [42] by generating execution cost-aware execution plans. Rocha-Junior et al. [41] propose a framework AGiDS based on a grid summary to capture the data distribution on each server, in order to reduce the amount of transferred data. Although we also use grid summary to optimize the pruning, the collected summaries and the processing strategies are distinct, due to the data uncertainty.

Moreover, Vlachou et al. [24] propose SKYPEER to solve the problem of subspace skyline queries in large-scale peer-to-peer networks, while in our work, we focus on the skyline queries in the full space. SKYPEER+ [38] further improves the thresholding scheme and routing in SKYPEER. Besides, Zhu et al. [27] propose a feedback-based distributed skyline (FDS) algorithm to compute skylines with economical bandwidth cost at the expense of several round-trips.

2.1.3. Approaches in other environments

Besides P2P environment, Xin et al. [31] propose an energy-efficient algorithm (SWSMA) to address the skyline query problem in wireless sensor networks. Relying on a distributed data stream model, Sun et al. [29] propose an algorithm (BOCS) for skyline queries over data streams.

Hose et al. [34] provide a comprehensive survey on skyline processing in highly distributed environments. Note that, all the aforementioned approaches focus on the processing distributed skylines over certain data, which can use the additivity principle [34] to address the skyline query problem. However, none of them can be used to deal with the skyline queries over distributed uncertain data.

2.2. Uncertain skyline queries

Currently, there is a growing body of work on answering skyline queries over uncertain data in a centralized manner [9–14,45] since its introduction [8]. To process the p-skyline that retrieves the objects whose probabilities of being skyline is greater than the threshold p, the Bottom-up and Top-down algorithms are proposed in [8] to optimize the query. Besides, all skyline query problem is extensively addressed in [11].

Moreover, Böhm et al. [12] study the continuous case of the probabilistic skyline query where each object is modeled as a mixture Gaussian distribution. To improve the query, all the objects can be indexed with the Gaussian tree [46] in the parameter space. Zhang et al. [13] extend the probabilistic skyline operator to data streams with a sliding window model. Li et al. [47] address the parallel skyline queries over uncertain data streams with sliding window partitioning and grid index. Besides, Lian and Chen [9] first introduces the concept of probabilistic reverse skyline and proposed algorithms for processing the query including the monochromatic and bichromatic fashions.

Uncertain skyline computation in distributed environments poses inherent challenges and requires non-traditional uncertain skyline techniques due to the distribution of content and lack of global knowledge. Ding et al. [32] first propose DSUD and its extension e-DSUD to process the distributed probabilistic skylines. The efficiency of the algorithms stems from a highly optimized feedback mechanism, where the central server transmits the precious information to each local site to prevent the delivery of a large number of unqualified skyline tuples. Nevertheless, the pruning capabilities of the algorithms are greatly limited by the local knowledge for pruning, which can be greatly alleviated by our proposed GDPS approach with the grid filtration and many other optimizations.

3. Preliminaries

In this section, we first provide some preliminaries related to our work, including the data uncertainty model, the system model and the problem definition.

3.1. Uncertainty model

Generally, the data in real world primarily includes *certain data* and *uncertain data*, and there is no strict definition or classification standards currently for the data. The Trio system [48] introduces various models to capture data uncertainty at different levels, where the data is classified into two types: *exact data* and *inexact data* [48]. There are many ways to describe inexact data, such as *uncertain data*, *probabilistic data*, *fuzzy data*, *approximate data*, *incomplete data*, *interval data* and *imprecise data*, etc.

According to the different granularities of uncertain entity, the data uncertainty can be specified by three levels: *group-based* (or *table-based*), *object-based* (or *tuple-based*) and *attribute-based* [49]. Generally, the following two kinds of uncertainty are extensively investigated in most of the studies:

1. **Tuple-level uncertainty**: also called *existential uncertainty*, which describes the probability for the presence or absence of the tuples. In most cases, the tuple independence assumption is used, in which the presence probabilities of different tuples are independent/disjoint of one another and mutual exclusivity may be defined to constrain the tuples in the database.
2. **Attribute-level uncertainty**: this type only refers to the uncertainties of the individual attributes, and usually includes two kinds of values: *fuzzy value* and *ambiguity value*. *Fuzzy* means that the value is difficult to be specified precisely, while *ambiguity* indicates that the value is randomly selected from many alternative values.

Moreover, according to the relationship of objects, two kinds of object-based model are frequently used with the *possible world* semantics: *independent model* and *general model* [50]. In the independent model, the objects are independent to each other and most of the existing studies on uncertain queries are assumed with this model to simplify the queries. While the general model is a general case, objects in which may be correlated, thus *generation rules* are often defined to describe the *exclusive* relationship of the objects.

Similar to the works [32,48,51,52], we also focus on the tuple-level data uncertainty with independent model of possible world semantics. Furthermore, the tuple uncertainty is represented by the discrete probability value by the form of $\langle x, 0.7\rangle$, where x denotes the tuple information, and 0.7 is the existential probability of the tuple, which is represented by a discrete value in $[0, 1]$.

Generally, for a possible world W, the probability of W appears is $P(W) = \prod_{t\in W} P(t) \times \prod_{t\notin W}(1 - P(t))$. Let Ω be the set of all possible worlds, then $\sum_{W\in\Omega} P(W) = 1$. Assume that $SKY(W)$ denote the set of elements in W that form the skyline of W, then the probability that e appears in the skylines of the possible worlds will be $P_{sky}(e) = \sum_{e\in SKY(W), W\in\Omega} P(W)$.

3.2. System model

Similar to the work [32], we focus on the tuple-level uncertainty with independent model of possible world semantics. Furthermore, the tuple uncertainty is represented by the discrete probability value by the form of $\langle A_1, A_2, \dots, A_n, p\rangle$, where $\langle A_1, A_2, \dots, A_n\rangle$ denotes the deterministic tuple information, and p denotes the existential probability of the tuple, which is represented by a discrete value within $[0, 1]$.

Moreover, we assume that the data are horizontally partitioned to all the local sites, and make no assumption on the existence of any overlay network that connects local sites in an intentional manner. Thus, a querying server (called *coordinator*) directly communicates with all the local sites, which is also the current commonly used processing framework for distributed skyline queries [19–21,27,32,41].

In the network, each local site owns part of the whole data. All the nodes including the coordinator and the local sites have their own capabilities for storing and processing. Moreover, all the tuples in each local site are independent and the tuples have the existent level uncertainty, whose uncertainty is represented by the discrete probabilities.

Generally, a good distributed skyline algorithm should achieve the following objectives:

- *Minimization of bandwidth consumption*. The communication overhead is usually the dominator factor due to the network delay and the economic cost associated with transmitting large amounts of data over a network when querying distributed uncertain data [19,32].
- *Progressiveness*. The algorithm should quickly return some early results soon after the beginning and produce a majority of the remaining results well before the end of the query [27].

Without loss of generality, as studied for many problems over distributed data, a critical requirement is to reduce the communication cost. Thus, our main objective of this paper is to retrieve the global q-skylines progressively by minimizing the total bandwidth consumption for the query, which is similar to the studies in [19–21,27,32,41].

3.3. Problem definition

Given a centralized data set D and a tuple $t \in D$ with existential probability $P(t)$. The skyline probability of a tuple t actually equals the probability that t exists and all the tuples that dominate t do not exist [32]. Thus, the skyline probability $P_{sky}(t, D)$ of tuple t against D can be calculated as:

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

Thus, we can further define the q-skyline as follows:

Definition 1 *(**q**-skyline).* Given a probability threshold q $(0 \le q \le 1)$, the q-skyline returns the set of uncertain objects, each of which takes a probability of at least q to be in the skyline set.

Based on the above definition, the problem of the distributed skyline query over uncertain data can be defined as follows:

Definition 2 *(Distributed probabilistic skyline query).* Given a set of m distributed sites $S = \{s_1, s_2, \dots, s_m\}$, each possesses an uncertain data set D_i $(1 \le i \le m)$ with size n_i, and a coordinator H is

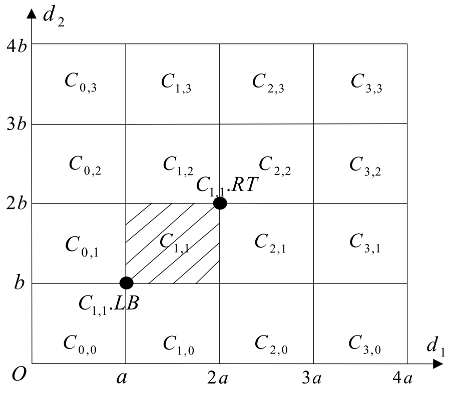


Fig. 2. An example of grid partitioning.

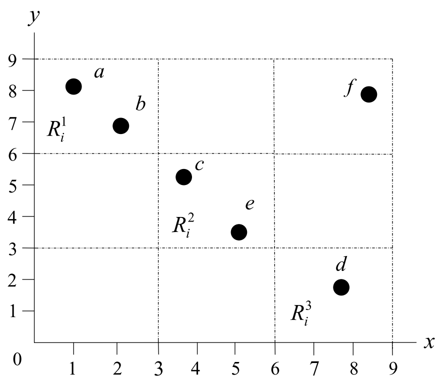


Fig. 3. The dominant relationships of cells.

responsible for scheduling and processing. The query retrieves the tuples whose global skyline probabilities are not smaller than a given threshold q $(0 \le q \le 1)$ from all the local sites at H. The global skyline probability $P_{g\_sky}(t)$ of the tuple t is defined as:

$$\begin{aligned} P_{g\_sky}(t) &= P(t) \times \prod_{t' \in D_1,\, t' \prec t} (1 - P(t')) \\ &\quad \times \prod_{t' \in D_2,\, t' \prec t} (1 - P(t')) \times \cdots \times \prod_{t' \in D_m,\, t' \prec t} (1 - P(t')) \\ &= P(t) \times \prod_{i=1}^{m} \Bigl(\prod_{t' \in D_i,\, t' \prec t} (1 - P(t')) \Bigr) \qquad (2) \end{aligned}$$

Assume that $D = D_1 \cup D_2 \cup \ldots \cup D_m$, then Eq. (2) can be further written as

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

Therefore, the goal of the query discussed in this paper is to retrieve the set of tuples from the global data set D with minimum bandwidth consumption, where each tuple t satisfies $P_{g\_sky}(t) \ge q$, and reduce the overall processing time as far as possible.

4. Grid summary

In this section, we first present the structure of the grid summary, including the grid partitioning, some grid-based definitions and the grid summary collection.

4.1. Grid partitioning

Since all the tuples share the same data space, we can divide the whole space into many cells. Thus, each tuple can find the cell that it belongs to. For instance, as shown in Fig. 2, the entire space is divided into many equal-sized cells, the width of the cell in each dimension is δ. Thus, the cell $C_{i,j}$ contains all the objects that satisfy $i \cdot \delta < e.d_1 \le (i+1) \cdot \delta$ and $j \cdot \delta < e.d_2 \le (j+1) \cdot \delta$. For simplicity, we denote an item e in $C_{i,j}$ as $e \in C_{i,j}$. On the contrary, for any object e, we can easily locate its corresponding cell $C_{i,j}$, where $i = \lfloor e.d_1/\delta \rfloor$ and $j = \lfloor e.d_2/\delta \rfloor$. Without loss of generality, we assume that smaller values are preferred in the skyline operator. Suppose that $C_{i,j}.LB$ and $C_{i,j}.RT$ denote the lower left corner and the upper right corner of the cell $C_{i,j}$, then for any item $e \in C_{i,j}$, we have $C_{i,j}.LB \prec e$ and $e \prec C_{i,j}.RT$.

Generally, a multi-dimensional tuple $t \in P_i$ in the region R_i^j (denoted as $P_i \in R_i^j$) means that for each dimension $d_i \in D$ and $p_i^j \in P_i$, $l_i^j \le p_i < u_i^j$. Assume that Num_i denotes the number of partitions in dimension i, the domain of each dimension is U, and the corresponding width of the cell is δ, then we have $Num_i = U/\delta$. Since the value domain of each dimension in this paper is $(0, 1)$ and each dimension has the same number of divisions, thus $Num_i = 1/\delta$ and the total number of cells is $(1/\delta)^d$.

4.2. Grid-based definitions

Without loss of generality, assume that minimum values are preferable. In this section, we introduce some definitions related to the grid summary for the query, which are similar to the definitions in [41].

Definition 3 *(Cell dominance).* If the right upper corner u^j of R_i^j dominates the left lower l^k of R_i^k, then R_i^j dominates R_i^k, which is denoted as $R_i^j \prec R_i^k$.

Definition 4 *(Partial cell dominance).* If R_i^j does not dominate R_i^k, but the left lower l^j of R_i^j dominates the right corner of R_i^k, then R_i^j partially dominates R_i^k, which is denoted as $R_i^j \rhd R_i^k$.

Definition 5 *(Cell incomparable).* If there is no point in R_i^j can dominate any point in R_i^k and vice versa, we call R_i^j and R_i^k are incomparable, which can be denoted as $R_i^j \sim R_i^k$.

As shown in Fig. 3, according to the above definitions, we can find that $R_i^2 \prec R_i^4$, $R_i^1 \rhd R_i^4$ and $R_i^1 \sim R_i^2$. Therefore, according to the definition of the skyline probability of a tuple t discussed in Section 3.3, we can easily draw the following conclusion for the local skyline probability $P_{l\_sky}(t)$ of tuple t as follows:

$$P_{l\_sky}(t) \le \begin{cases} \prod_{u \in R_i^j,\, u \prec t} (1 - P(u)), & \text{if } t \in R_i^j, \\ \prod_{s \in R_i^k} (1 - P(s)), & \text{if } R_i^j \prec R_i^k, \\ \prod_{r \in R_i^k,\, r \prec t} (1 - P(r)), & \text{if } R_i^j \sim R_i^k \end{cases} \qquad (4)$$

Assume that the whole space are partitioned into n^d cells, then we can rewrite the local skyline probability of tuple t in site S_i located at grid cell R_i^j as follows:

$$\begin{aligned} P_{l\_sky}(t) &= P(t) \times \prod_{t \in D_i,\, t' \prec t} (1 - P(t')) \\ &= P(t) \times \prod_{x=1,\, x \ne j,\, R_i^x \sim R_i^j,\, r \in R_i^x,\, r \prec t}^{n^d} (1 - P(r)) \end{aligned}$$

$$\times \prod_{x=1,\, x\neq j,\, R_i^x \prec R_i^j,\, s\in R_i^x}^{n^d} \big(1-P(s)\big)$$
$$\times \prod_{u\in R_i^j,\, u\prec t} \big(1-P(u)\big) \tag{5}$$

To facilitate the illustration of the grid summary, we also provide the following two definitions.

Definition 6 *(Local cell non-existing probability, lnep).* The *lnep* value of the cell R_i^j means the probability that all the tuples do not exist in the cell R_i^j, which can be defined as:

$$P_{lnep}\big(R_i^j\big) = \prod_{t\in R_i^j} \big(1-P(t)\big) \tag{6}$$

Definition 7 *(Global cell non-existing probability, gnep).* Given m local sites, the *gnep* means the global probability that all the tuples do not exist in the cell R_i^j, which can be defined as:

$$P_{gnep}\big(R_i^j\big) = \prod_{s=1}^{m}\Big(\prod_{t\in R_i^j}\big(1-P(t)\big)\Big) \tag{7}$$

According to Eqs. (5), (6) and (7), we can use the cell non-existing probability (*lnep* and *gnep*) instead of computing the probabilities of all the tuples to improve the computation in GDPS, which will be described in detail in Section 5.

4.3. Grid summary collection

Suppose that we want to map the local data set D_i to the local grid summary $G(D_i)$. When mapping D_i to $G(D_i)$, each tuple t in D_i is mapped to a certain cell $c(t)$. We use $c.CS(D)$ to denote the set of tuples in D mapped to cell c, which is associated with a cell probability $c.cp$, which represents the probability that none of the tuples in $c.CS(D)$ exist, i.e.,

$$c.cp = \prod_{t\in c.CS(D)} \big(1-P(t)\big) \tag{8}$$

Since $c.CS(D) = c.CS(D_1) \cup c.CS(D_2) \cup \ldots \cup c.CS(D_m)$, thus we have

$$c.cp = \prod_{t\in c.CS(D)} \big(1-P(t)\big) = \prod_{i=1}^{m}\Big(\prod_{t\in c.CS(D_i)} \big(1-P(t)\big)\Big) \tag{9}$$

From the above formula, we can see that the probability actually equals the *gnep* value of the cell, i.e.,

$$c.cp = \prod_{t\in c.CS(D)} \big(1-P(t)\big) = P_{gnep}(c) = \prod_{i=1}^{m} P_{lnep}(c) \tag{10}$$

Therefore, we can first compute all the *lnep* values for all the cells locally in each site, and then transmit them to the coordinator. Thus, we can further get the *gnep* values of the cells based on the *lnep* values with Eq. (10). Note that, the obtained *gnep* information is the key to our query pruning, which will be explained in the next section. For brevity, the frequently used symbols in the paper are summarized in Table 2.

Table 2
Frequently used symbols.

Symbol	Meaning
H	The central coordinator server
m	The number of local sites
d	The dimensionality of the tuples
q	The probability threshold
n	The number of partitions per dimension
LS_i	The i-th local site
D_i	The data set in LS_i
D	The global uncertain database
L	The priority queue maintained by H
D_L	The set of data maintained in L
SKY_H	The set of global skylines obtained in H
PRT_H	The set of tuples pruned in H
CAN_{D_i}	The candidate skylines in D_i
$P_{g\_sky}(t)$	Global skyline probability of object t
$P_{l\_sky}(t, D_i)$	Local skyline probability of t against D_i
$P_{t\_sky}(t, D_i)$	Temporary skyline probability of t against D_i
$P_{sky}(t, D_i)$	Skyline probability of t against D_i

5. The general framework of GDPS

The general framework of GDPS mainly consists of the *preprocessing* part and the *iteration process* part. In the first part, we try to prune the unqualified skylines early with our proposed grid summary. In the second part, we iteratively compute the global skylines with an optimized feedback mechanism.

5.1. Preprocess with grid filtration

The preprocessing part in GDPS consists of the following five phases before the iteration process, which is definitely distinct from the e-DSUD algorithm.

- **Grid-partition phase:** each site splits the data space into cells with a uniform metric, thus all the tuples can find the cells that they belong to;
- **Grid-sending phase:** each local site computes the local cell non-existing probability (*lnep*) of each cell, and then sends the grid summaries to H;
- **Sever-calculation phase:** H computes the global non-existing probability (*gnep*) of each cell and labels all the cells with the probabilities;
- **Grid-broadcast phase:** H broadcasts all the processed grid summaries including the probabilities used for grid filtration to all the local sites;
- **Grid-filtration phase:** after receiving all the information from H, each local site prunes all the unqualified tuples based on the *gnep* of each cell.

As shown in Fig. 4, each local site first maps the tuples into different cells and sends the *lnep* values to H, in order to compute the *gnep* values for pruning the unqualified tuples in the following processes.

According to the grid summary discussed in Section 4.3, we can label all the cells based on the following rule in the *Sever-Calculation* phase: If the cell is dominated by a cell whose *gnep* is less than q, then the cell is labeled with "0", which means that the tuples in the cell are unqualified to be the global skylines; Otherwise, labeled with "1". Thus, we can rapidly prune all the tuples which are labeled with "0". For example, R_{32} and R_{33} in Fig. 5 are labeled with "0", since they are dominated by R_{21} with $lnep < q$.

The detailed description of the preprocessing with grid filtration is shown in Algorithm 1. Note that, the grid summary GS in step 8 mainly contains the *gnep* values and the labeled values of all the cells. Besides, the local skyline probability $P_{l\_sky}(t, D_i)$ in step 12 can be computed by $P(t) \times \prod_{t'\in D_i, t'\prec t}(1-P(t'))$, while

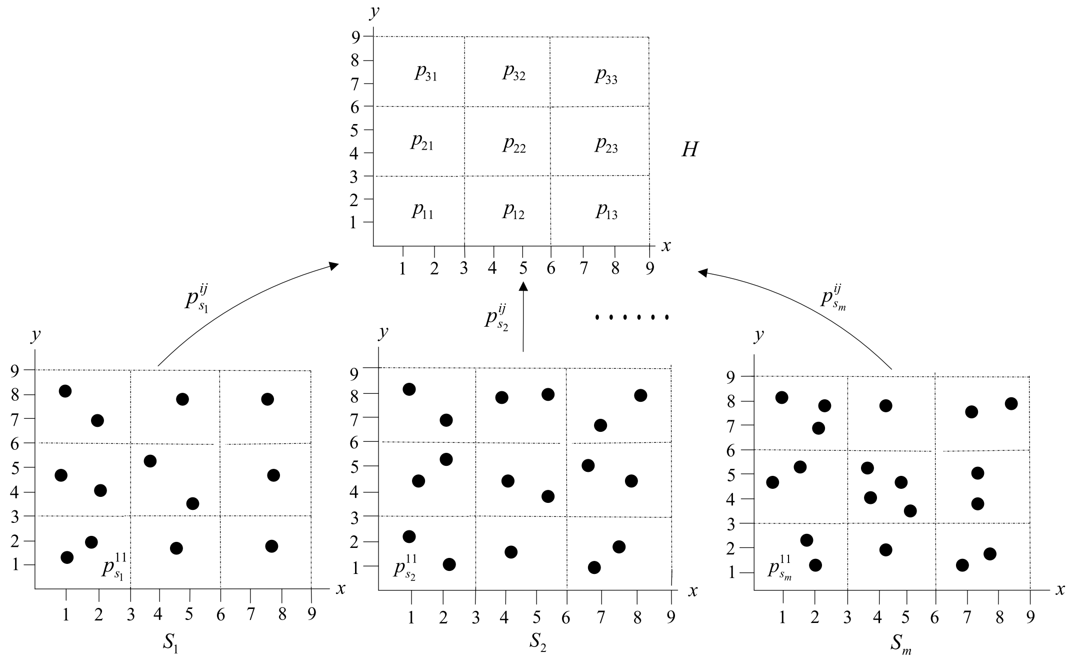


Fig. 4. The processing based on grid partition.

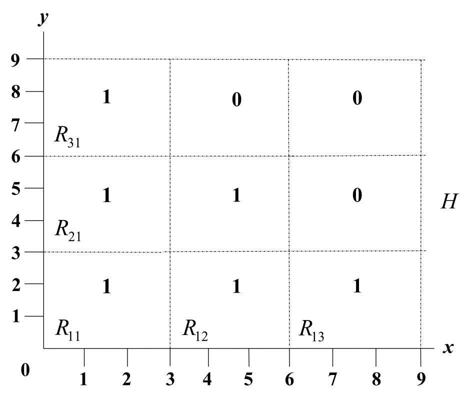


Fig. 5. The pruning region with grid summary.

the temporary skyline probability $P_{t\_sky}(t, D_i)$ of tuple t as mentioned in step 13 can be obtained as follows:

$$P_{t\_sky}(t, D_i) = P(t) \times \prod_{R_g \prec t} \left(\prod_{t' \in R_g} \left(1 - P(t')\right) \right) \tag{11}$$

To summarize, in the preprocessing part we collect the global knowledge of all the tuples with the grid summary and prune the tuples that cannot be the global skylines as far as possible.

5.2. Iteration with optimized feedback

As shown in Algorithm 2, GDPS mainly consists of two parts: the preprocessing with grid filtration, and the iteration process with feedback. The step 2 and steps 7∼8 correspond to the two parts, respectively.

After the preprocessing, we can get the first candidate set CAN_{D_i} $(1 \le i \le m)$ in each local site, and then we select a representative tuple t^i_{max} that is considered as the most likely to be the skyline with a certain tuple selecting strategy, which will be discussed in detail later. After receiving all the representative tuples from all the local sites, H starts the iteration process.

Algorithm 1: *Preprocessing*(D, LS, CAN): Preprocess with grid filtration.

```
Input  : all the local databases: $D = \{D_i | 1 \le i \le m\}$;
         all the local sites: $LS = \{LS_i | 1 \le i \le m\}$;
         the probability skyline threshold: $q$
Output: $CAN = \{CAN_{D_i} | 1 \le i \le m\}$
1  begin
2      $H$ sends the partition metric to $LS$;
3      foreach $LS_i$ do
4          Map tuples in $D_i$ to cells with the metric;
5          Compute the $lnep$ values for all the cells;
6          Send all the $lnep$ values to $H$;
7      $H$ collects all the $lnep$ values and computes the $gnep$ value of each cell
       with Eq. (7);
8      $H$ labels each cell with $gnep$ and obtain the grid summary $GS$;
9      $H$ sends the $GS$ to all the local sites $LS$;
10     foreach $LS_i$ do
11         Prune all the unqualified tuples with $GS$;
12         Calculate $P_{l\_sky}(t, D_i)$ values for all tuples;
13         Compute $P_{t\_sky}(t, D_i)$ values for all tuples;
14         Obtain $CAN_{D_i}$ by filtering the unqualified tuples with $P_{l\_sky}(t, D_i)$
           and $P_{t\_sky}(t, D_i)$;
```

Algorithm 2: GDPS.

```
1  begin
2      $Preprocessing(D, LS, CAN)$;
3      foreach $LS_i$ do
4          Select the tuple $t^i_{max}$ from $CAN_{D_i}$ with a certain strategy;
5          Send $t^i_{max}$ to the coordinator $H$;
6      $H$ collects the $t^i_{max}$ $(1 \le i \le m)$ from all the local sites as the set $D_L$;
7      while $D_L \ne \emptyset$ do
8          $IterativeFeedback(D_L, LS, SKY_H)$;
9      return $SKY_H$
```

In the iteration process part of GDPS, we attempt to further prune the unqualified tuples from the candidate skyline sets to reduce the number of tuples that need to be transmitted.

As shown in Algorithm 3, we try to update the temporary skyline probability $f(t, D_x) = P_{t\_sky}(t, D_x)$ of tuple t that belongs to

Algorithm 3: *IterativeFeedback*(D_L, LS, SKY_H): Iterative processing with feedback.

```
Input: all the local sites: $LS = \{LS_i | 1 \le i \le m\}$;
       the set of $t_{max}$s from all the local sites:
       $D_L = \{t^i_{max} | 1 \le i \le m\}$;
       the candidate sets: $CAN_{D_i}$ $(1 \le i \le m)$
Output: final global skyline set $SKY_H$
1  begin
2      Update $P_{t\_sky}(t, D_L)$ for all tuples in $D_L$;
3      Filter unqualified tuples $PRT_H$ from $D_L$;
4      $H$ pops the tuple $t_M$ with the biggest $P_{t\_sky}(t, D_L)$ value from $D_L$;
5      $H$ sends $t_M$ to each local site except the site $LS_t$ that contains $t_M$;
6      foreach $LS_i$ (except $LS_t$) do
7          Compute the dominance probability of $t_M$ with
           $P^i_{dop}(t) = \prod_{t' \in D_i, t' \prec t}(1 - P(t'))$;
8          Send the $P^i_{dop}(t)$ value to coordinator $H$;
9          Prune the tuples from $CAN(i)$ with $t_M$;
10     $H$ computes the $P_{g\_sky}(t_M)$ value with Eq. (3) based on the received
       values;
11     if $P_{g\_sky}(t_M) \ge q$ then
12         add $t_M$ to the global skyline set $SKY_H$;
13     Fetch new tuples $N_H$ from $LS_i$ and sites that include $PRT_H$;
14     Update new candidate set in $H$ with $D_L = D_L \cup N_H$;
```

data set D_x in step 2 according to the following rule:

$$f(t, D_L) = \begin{cases} f(t, D_i) \times \prod_{t' \in D_L, t' \prec t}(1 - P(t')), & \text{if } t \in N_H, \\ f(t, D_L) \times \prod_{t' \in N_H, t' \prec t}(1 - P(t')), & \text{otherwise} \end{cases} \quad (12)$$

Thus, in step 3 we can safely prune the tuples with $P_{t\_sky}(t, D_L) < q$ from D_L. Furthermore, in step 3 we can further filter the unqualified tuples with the following corollary, which has been proved in [32].

Corollary 1. *The global skyline probability $P_{g\_sky}(t)$ of tuple t from site S_i is upper bounded by its local skyline probability multiplies the upper bound of its local skyline probabilities to other databases D_x, i.e.,*

$$\begin{aligned} P_{g\_sky}(t) = \prod_{x=1}^{m} P_{sky}(t, D_x) &\le P_{sky}(t, D_i) \\ &\times \prod_{x=1, x \ne j, t \in D_x, s \in L, s \prec t}^{m} \frac{P_{sky}(s, D_x)}{P(s)} \times \big(1 - P(s)\big) \\ &= P_{g\_sky}(t)^* \end{aligned} \quad (13)$$

In addition, steps 4–10 aim to get the global skyline probability for tuple t_M. After completing all above operations, H fetches a batch of new tuples from all the local sites and starts a new iteration. Specifically, step 9 prunes all the candidates in each local site according to the updated local skyline probability $P(t)$ against the feedback tuple t_M, i.e.,

$$P(t) = \begin{cases} P_{l\_sky}(t, D_i) \times (1 - P(t')), & \text{if } t' \prec t \\ P_{l\_sky}(t, D_i), & \text{otherwise} \end{cases} \quad (14)$$

After updating the probabilities, we can filter all the unqualified tuples and reorder the new candidates with the updated local skyline probabilities. The GDPS approach does not terminate until $D_L = \emptyset$.

Note that, the reason we adopt grid summary primarily come from two considerations: (1) we can easily get the grid summary as illustrated in Section 4.3, and (2) we can optimize the computation of the dominance probability of the tuples returned from the coordinator, which corresponds to the step 7 in Algorithm 3. Thus, the computational time can be greatly reduced.

6. Further optimization

In this section, we further optimize the query from the following aspects: (1) pruning optimization in the local sites, (2) the tuple selecting strategy in each local site, and (3) pruning optimization in the coordinator.

6.1. Local pruning optimization

As discussed in Section 5.2, we prune the local candidates with the updated local skyline probability $P(t)$. This strategy is not efficient since we cannot make use of all the feedback tuples in former iterations. Instead, we can adopt the updated temporary skyline probability $P_{t\_sky}(t, D_i)$ to filter the local candidates in each local site according to the following observation.

Observation 1. Assume that site S_i receives the feedback tuples from all the other sites are the sets of DF_j $(1 \le j \le m \wedge i \ne j)$, and the total returned set of tuples is $DR_i = DF_1 \cup DF_2 \cup \cdots \cup DF_m$, where $DF_j \subseteq D_j$ $(1 \le j \le m \wedge j \ne i)$. Therefore, we have

$$\begin{aligned} P_{g\_sky}(t) &= \prod_{i=1}^{m} P_{sky}(t, D_i) \\ &\le P_{sky}(t, D_i) \times \prod_{j=1, j \ne i}^{m} P_{sky}(t, DF_j) \\ &= P_{sky}(t, D_i) \times \prod_{j=1, t' \in DF_j, t' \prec t}^{m} \big(1 - P(t')\big) \\ &= P_{sky}(t, D_i) \times P_{sky}(t, DR_i) \\ &= P_{t\_sky}(t, D_i) \end{aligned} \quad (15)$$

It must be noted that, the $P_{t\_sky}(t, D_i)$ value of tuple t is updated constantly when receiving a new feedback tuple from H with the following rule:

$$P_{t\_sky}(t, D_i)* = \begin{cases} (1 - P(t')), & \text{if } t' \prec t \\ 1, & \text{otherwise} \end{cases} \quad (16)$$

6.2. Tuple selecting optimization

Based on the feedback strategy, one important problem needs to be addressed: how to choose the "best" tuples from each local site to the coordinator H? For this problem, we have designed four typical tuples selecting strategies, which are described as follows:

- *Max-Temp*: choose the tuple t with the largest updated temporary skyline probability $P_{t\_sky}(t, D_i)$.
- *Max-Local*: choose the tuple t with the maximum updated local skyline probability, which can be calculated with Eq. (14).
- *Min-Rank*: select the tuple t with the minimum sum of rank value *RankScore* of t.
- *Max-Domin*: select the tuple t with the maximum dominance region VDR_t of t.

Note that, the *RankScore* in the *Min-Rank* strategy means that the sum of the all the ranks for all the dimensions. Assume that the rank of the value in dimension i is r_i, then the *RankScore* can be computed by $\sum_{i=1}^{d} r_i$, where d is the dimensionality of the tuple.

The dominance region for tuple t is achieved with formula $VDR_t = \prod_{k=1}^{d}(max_k - t.k)$, where max_k is a pre-defined value larger than the global domain upper bound b_k, and $t.k$ is the value of dimension k of tuple t, which is similar to [30]. Assume that the value range of the dimension k is $[s_k, b_k]$ in the data space, then the volume of tuple t's dominance region is defined as $VDR_t = \prod_{k=1}^{d}(b_k - t.k)$.

In this paper, we evaluate all above tuple selecting strategies in Section 7.2, and we can find that the performance of the *Max-Temp* strategy is relatively better than the others.

6.3. Server pruning optimization

Since part of the tuples in L may be filtered in the iteration part, and the pruned tuples will not be transmitted to the local sites. If we make use of these tuples for pruning, more tuples in L can be further filtered. Thus, we can transmit the updated temporary skyline probability to H for further optimization. In particularly, we have the following observation.

Observation 2. Assume that PRT_H denotes the set of tuples pruned in L accumulated in all the iterations, and $PRT_H \backslash D_i$ denotes the set of tuples in PRT_H that do not from site S_i. For any tuple t in L from database D_i at local site S_i, the upper bound of the global skyline probability of t will be bounded as:

$$\begin{aligned} P_{g\_sky}(t) &= \prod_{i=1}^{m} P_{sky}(t, D_i) \le P_{t\_sky}(t, D_i) \times P_{sky}(t, D_L) \\ &\quad \times \prod_{t' \in PRT(H) \wedge t' \notin D_i} \left(1 - P\left(t' \prec t\right)\right) \\ &= P_{t\_sky}(t, D_i) \times P_{sky}(t, D_L) \times P_{sky}(t, PRT_H \backslash D_i) \qquad (17) \end{aligned}$$

The correctness of Eq. (17) is rooted from that, the tuples used for computing $P_{t\_sky}(t, D_i)$, $P_{sky}(t, D_L)$ and $P_{sky}(t, PRT_H \backslash D_i)$ are independent and not reduplicate. Specifically, if we denote the set of tuples that have already been used for updating $P_{t\_sky}(t, D_i)$ in site S_i as TS_{D_i}, then we can easily find that $TS_{D_i} \cup D_L \cup PRT_H \backslash D_i \subseteq D$ and $TS_{D_i} \cap D_L \cap PRT_H \backslash D_i = \emptyset$. Note that, TS_{D_i} includes all the tuples of D_i and all the feedback tuples till now.

Based on the above observation, we can further prune the candidate tuples obtained by the step 2–3 in Algorithm 3. Thus, in the iteration process phase, we only need to maintain all the pruned tuples in a priority queue in coordinator H, and sort all the tuples with the descending order of the updated temporary skyline probabilities. With these maintained tuples, more candidates can be further pruned. Moreover, for the purpose of reducing the computation and the storage cost, we can only maintain a fixed part number of the pruned tuples and replace the tuples with a certain heuristic strategy.

7. Experimental evaluation

In this section, we present results of a comprehensive performance study to evaluate the effectiveness and efficiency of the proposed techniques.

7.1. Experimental setup

In our experiments, we evaluate the approaches on both synthetic data sets and real data sets.

Synthetic data sets. The synthetic data sets we used are created with two popular distributions [7], i.e., Independent and Anti-correlated, whose generation follows the description in Fig. 6. The overall cardinalities of the generated data sets are 2 million and 10 million tuples. Moreover, we use uniform distribution to randomly assign an occurrence probability to each tuple, where the mean value and the standard deviation equal 0.7 and 0.3, respectively.

Table 3
System parameters.

Parameter	Values
$\lvert D\rvert$	2×10^6 (2M) 10×10^6 (10M)
m	40 **60** 80 100
d	2 **3** 4 5
q	**0.3** 0.5 0.7 0.9
n	2 6 **10** 14 18 22 26

Real data sets. The real data sets adopted in our experiments from two aspects: (1) The data set (Zillow) collected from www.zillow.com that contains information about real estate all over the United States. We deal with at most four dimensions namely number of bedrooms and bathrooms, the built year and the price gap, where the price gap of a house is computed as the maximum house price in the data set minus the price of the house. Thus, this type of data is more likely Anti-correlated data, as a large room tends to has small price gap. (2) The data set (IPUMS) downloaded from www.ipums.org that contains the census information of population, including demographic, geographic, household, income, consumption, etc. Since the attributes have few correlations among them, the data set is more likely an independent data set. Note that, the cardinality of the above two data sets are both 2×10^6 (2M). Moreover, we associate uncertainty to the data by randomly assigning each tuple with an existential probability following the normal distribution, whose the mean value μ equals 0.75 and the standard deviation σ equals 0.25.

We conduct all our experiments with real deployment on a data center, where each physical host is configured with dual-core Intel 2.6 GHz Xeon CPU, 4 GB main memory and a 1 TB hard disk, gigabit ethernet. Moreover, each host is deployed with two virtual machines, each of which corresponds to a local site in our experiments. All the evaluated approaches including GDPS, GBPS, DSUD and e-DSUD are all implemented in C++ running on the CentOS operating system.

Assume that there exist m local sites, the total number of tuples is $|D|$, the probability threshold that user specified is q, and each local site has the same number of tuples $|D|/m$. In the following experiments, we mainly evaluate the proposals in terms of bandwidth consumption, i.e., the number of tuples transmitted over the network. Table 3 summarizes the parameters as well as their values to be examined, where all the parameters take default values as indicated in bold unless otherwise specified.

In the experiments, we first evaluate the performance of various tuple selecting strategies, and then we evaluate the efficiency of our proposed approach GDPS against dimensionality d, number of the local sites m, the probability threshold q and the partition number per dimension n under the Independent and Anti-correlated data sets. Finally, we evaluate the proposals under real data sets.

7.2. Performance with tuple selecting strategies

Firstly, we evaluate four strategies discussed in Section 6 against the dimensionality d, the number of the local sites m, and the probability threshold q, under the Independent and Anti-correlated data sets with cardinality 10M, respectively.

As shown in Figs. 7 and 8, the performance of the *Max-Domin* strategy is the worst, whereas that of the *Max-Temp* strategy is the best, under the Independent and Anti-correlated data sets with all our considered parameters. The distinct difference between the results of these two strategies probably stems from the different skyline querying definitions. In the context of our probabilistic skyline query, the pruning ability of one tuple relies on not only the volume of a tuple's dominance region, but also the existential probability of the tuple compared to the traditional skyline queries.

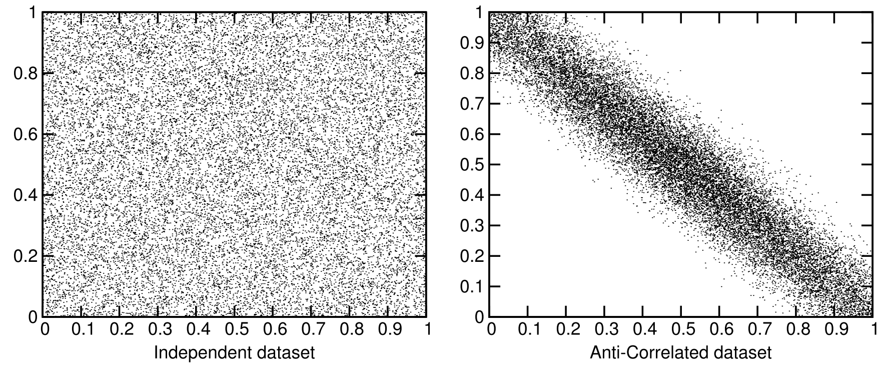


Fig. 6. An illustration of the synthetic data set.

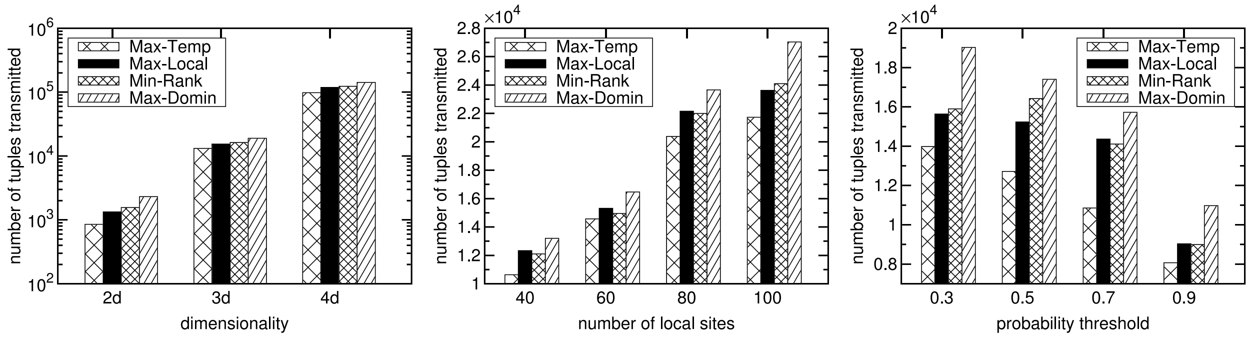


Fig. 7. Performance *vs.* Independent data sets.

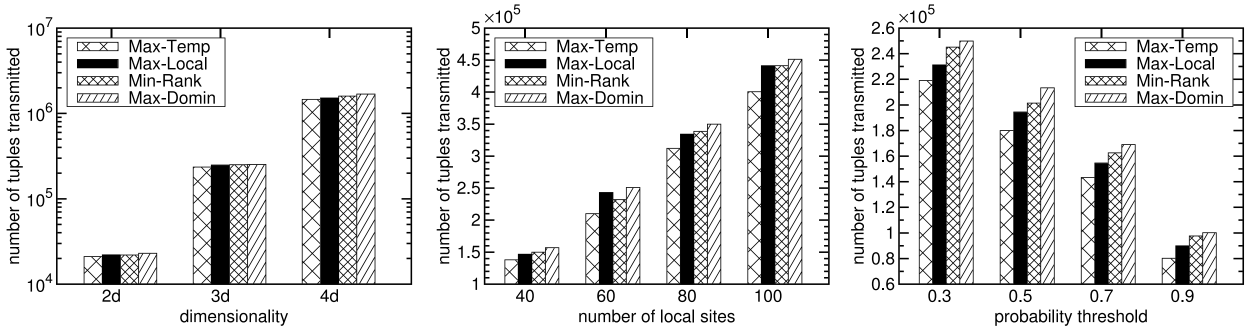


Fig. 8. Performance *vs.* Anti-correlated data sets.

Specially, we believe that the superiority of the *Max-Temp* strategy stems from the similar way of computing the $P_{t\_sky}(t, D_i)$ and $P_{g\_sky}(t)$ accumulatively.

Since the $P_{t\_sky}(t, D_i)$ value in the *Max-Temp* strategy is calculated for pruning the unqualified tuples in local sites, we can directly use it to select tuples to the coordinator H. Since it can not only save the computation cost, but also get a better performance, we adopt the *Max-Temp* strategy in our GDPS approach to evaluate the performances.

7.3. GDPS performance with parameters

7.3.1. Performance with dimensionality

As shown in Fig. 9, the total bandwidth consumption increases as we vary d from 2 to 4 under the Independent and Anti-correlated data sets, respectively. This is as expected, as the larger number of dimensionality would make more tuples not to be dominated by others, which would make the final skyline set become larger. Obviously, e-DSUD and GBPS require less bandwidth than DSUD, and GDPS requires the least communication cost. Furthermore, we also observe that the Anti-correlated data sets always have the larger bandwidth consumption than the Independent under the same experimental settings, which is similar to the situations on centralized data sets.

7.3.2. Performance with number of local sites

Fig. 10 shows the total bandwidth consumption when we vary m from 40 to 100, under the Independent and Anti-correlated data sets. The number of transmitted tuples increases as m getting larger. Since the total number of final skyline tuples that should be delivered from H is fixed according to the data sets. Thus, the larger of the number of the local sites is, the more bandwidth consumption will be. Moreover, GDPS requires the least bandwidth consumption as shown in Fig. 10, which indicates the efficiency of our proposed grid filtration mechanism in GDPS.

7.3.3. Performance with probability threshold

As shown in Fig. 11, the number of transmitted tuples for all the approaches decreases as q getting larger. This is as expected, as the probability threshold affects the total size of the final skylines. Generally, the smaller threshold is, the larger number of the skylines will be. The reason is that if a tuple t belongs to q-skyline, then it will be always in the result of q'-skyline if $q' \leq q$. Thus, the total number of final skylines delivered over the network decreases

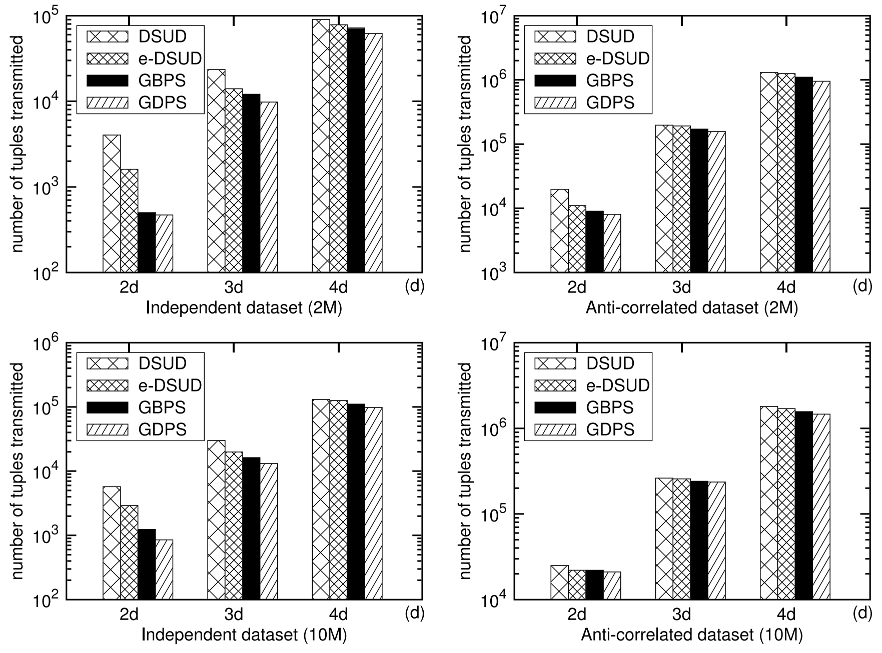


Fig. 9. Performance *vs.* dimensionality d.

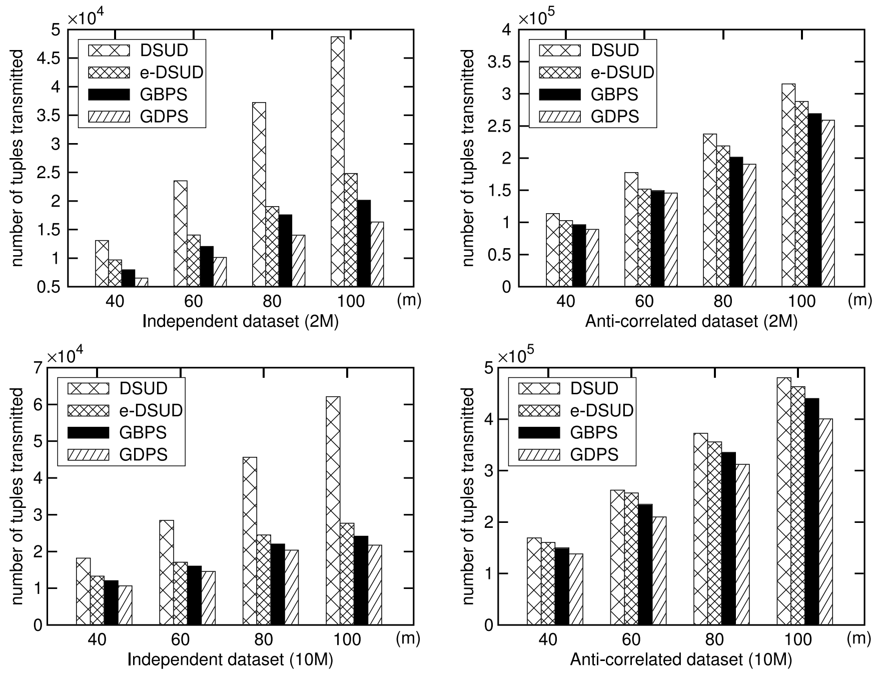


Fig. 10. Performance *vs.* number of local sites m.

according to the increase of q. Note that, the query performance is very sensitive to the variation of probability threshold, since the grid filtration and feedback mechanism used in GDPS can prune most of the unqualified skyline candidates with larger threshold. Moreover, we can find that in Anti-correlated data set the improvement is limited as most of cells contain few tuples, which limits the pruning ability.

7.3.4. *Performance with grid partition number*

In this experiment, we evaluate the effect on the grid partition. As shown in Fig. 12, the numbers of transmitted tuples decreases firstly and then towards to steady when we increase the partition number per dimension n from 2 to 26. The reason for this phenomenon is that, if the number of cells is small, then the number of tuples in each cell is large, which results in the coarse grid partition and the pruning efficiency is not obvious. On the contrary, the space partition is more concise, and the filtration is more efficient. Note that, the total cell number exponential increases as we increase n, which may easily result in the large number of cells, and increase the computation cost correspondingly. Consequently, in order to balance the communication cost and computation cost, we must choose an appropriate number of n.

7.4. *Evaluation with real data sets*

As illustrated in Section 7.1, we adopt the Zillow and IPUMS data sets to evaluate the four approaches against dimensionality d, number of the local sites m and the probability threshold q.

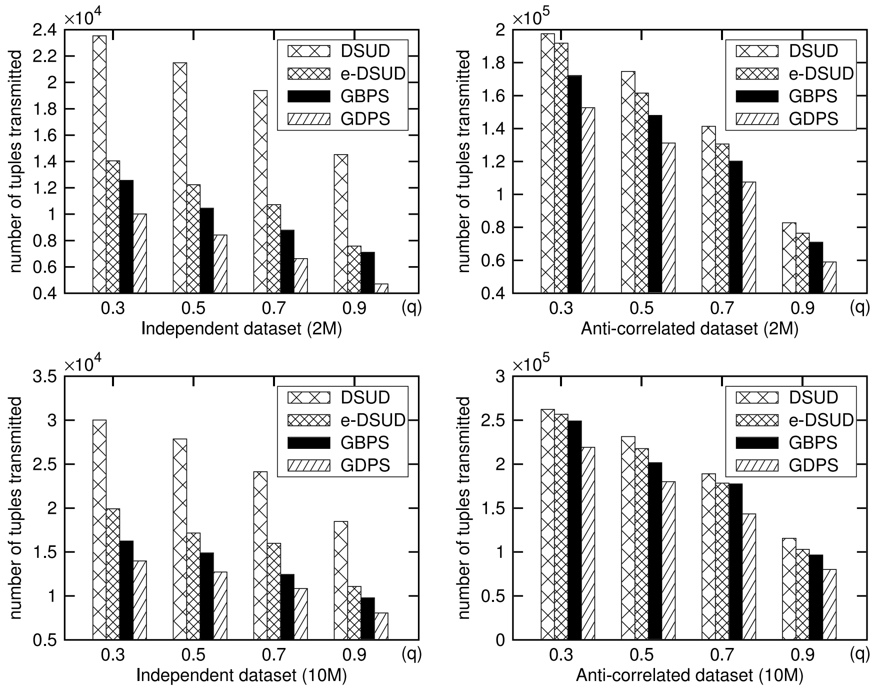
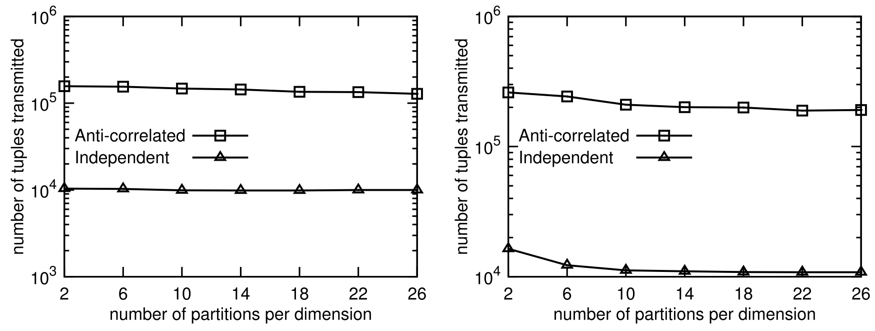
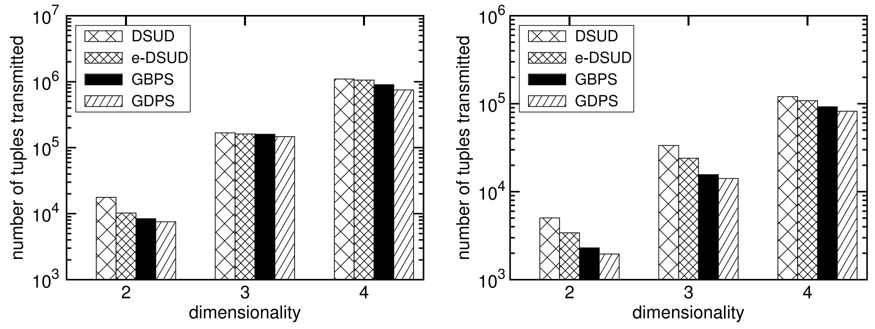


Fig. 11. Performance *vs.* threshold *q*.



(a) 2M data set (b) 10M data set

Fig. 12. Performance *vs.* grid partition.



(a) Zillow data set (b) IPUMS

Fig. 13. Performance *vs.* dimensionality with real data.

From Figs. 13, 14 and 15, we can see that the results are also similar to the results on synthetic data set discussed before, where the Zillow data set always have the larger bandwidth consumption than the IPUMS data set under the same experimental settings. Furthermore, we observe that the bandwidth consumption for the Zillow data set approaches to the Anti-correlated data set, while the results of the IPUMS are more close to the Anti-correlated data set. Nevertheless, the performances of GDPS and GBPS are real relative better than DSUD and e-DSUD, and GDPS is a little better than GBPS. Therefore, we can conclude that the proposed approach GDPS can get better performance compared with other approaches.

8. Conclusions

In this paper, we have addressed the problem of skyline queries over distributed uncertain data sets. To accelerate the query pro-

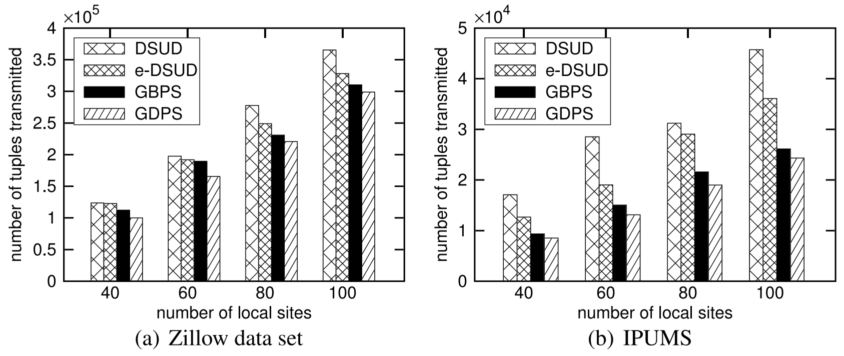


Fig. 14. Performance *vs.* number of local sites with real data.

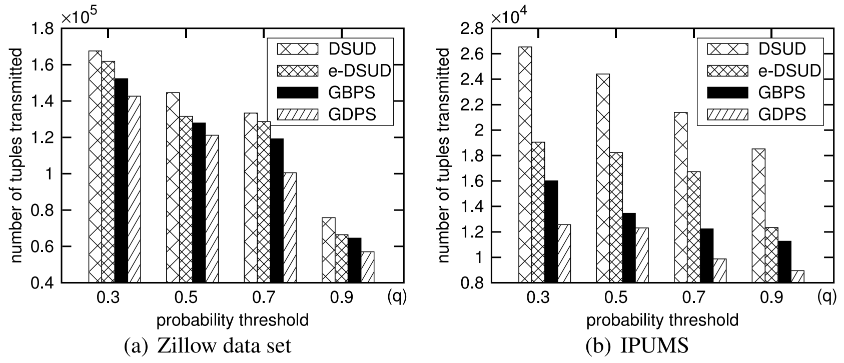


Fig. 15. Performance *vs.* threshold with real data.

cessing, we propose an efficient pruning mechanism for preprocessing with grid summary. Furthermore, we propose many strategies for optimizing the queries based on a feedback mechanism. Extensive experimental results with real data and synthetic data have verified the effectiveness and efficiency of the proposals. In our future work, we will consider querying the skylines over complex distributed uncertain data streams, as there are many potential demands for continuous skyline query currently [53].

Acknowledgements

This work was supported by the National Grand Fundamental Research 973 Program of China (Grant No. 2011CB302601), the National Natural Science Foundation of China (Grant No. 61379052), the National High Technology Research and Development 863 Program of China (Grant No. 2013AA01A213), the Natural Science Foundation for Distinguished Young Scholars of Hunan Province (Grant No. 14JJ1026), Specialized Research Fund for the Doctoral Program of Higher Education (Grant No. 20124307110015).

References

[1] C. Ré, N. Dalvi, D. Suciu, Efficient top-*k* query evaluation on probabilistic data, in: Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE), IEEE, 2007, pp. 886–895.

[2] T. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, P. Shenoy, Probabilistic inference over RFID streams in mobile environments, in: Proceedings of 25th IEEE International Conference on Data Engineering (ICDE), 2009, pp. 1096–1107.

[3] R. Gupta, S. Sarawagi, Creating probabilistic databases from information extraction models, in: Proceedings of the International Conference on Very Large Data Bases (VLDB), 2006.

[4] L. Chen, M. Özsu, V. Oria, Robust and fast similarity search for moving object trajectories, in: Proceedings of the ACM International Conference on Management of Data (SIGMOD), 2005, pp. 491–502.

[5] J. Pei, M. Hua, Y. Tao, X. Lin, Query answering techniques on uncertain and probabilistic data: tutorial summary, in: Proceedings of ACM SIGMOD, 2008, pp. 1357–1364.

[6] X. Sun, M. Li, H. Wang, A family of enhanced (l, α)-diversity models for privacy preserving, Future Gener. Comput. Syst. 27 (2011) 348–356.

[7] S. Börzsönyi, D. Kossmann, K. Stocker, The skyline operator, in: Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2001, pp. 421–430.

[8] J. Pei, B. Jiang, X. Lin, Y. Yuan, Probabilistic skylines on uncertain data, in: Proceedings of the International Conference on Very Large Data Bases (VLDB), 2007, pp. 15–26.

[9] X. Lian, L. Chen, Monochromatic and bichromatic reverse skyline search over uncertain data, in: Proceedings of the ACM International Conference on Management of Data (SIGMOD), 2008, pp. 213–226.

[10] C. Böhm, F. Fiedler, A. Oswald, C. Plant, B. Wackersreuther, Probabilistic skyline queries, in: Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), 2009, pp. 651–660.

[11] M. Atallah, Y. Qi, Computing all skyline probabilities for uncertain data, in: Proceedings of the ACM Symposium on Principles of Database Systems (PODS), 2009, pp. 279–287.

[12] B. Christian, F. Frank, O. Annahita, Computing all skyline probabilities for uncertain data, in: Proceedings of the IEEE International Conference on Data Mining (CIKM), ACM, 2009.

[13] W. Zhang, X. Lin, Y. Zhang, W. Wang, J. Yu, Probabilistic skyline operator over sliding windows, in: Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2009, pp. 1060–1071.

[14] M. Atallah, Y. Qi, H. Yuan, Asymptotically efficient algorithms for skyline probabilities of uncertain data, ACM Trans. Database Syst. 36 (2) (2011) 12.

[15] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, W. Hong, Model-driven data acquisition in sensor networks, in: Proceedings of the International Conference on Very Large Data Bases (VLDB), 2004.

[16] M. Alicherry, T. Lakshman, Network aware resource allocation in distributed clouds, in: Proceedings of the IEEE International Conference on Computer Communications (INFOCOM), 2012, pp. 963–971.

[17] K. Deng, X. Zhou, H. Shen, Multi-source skyline query processing in road networks, in: Proceedings of the IEEE International Conference on Data Engineering (ICDE), IEEE, 2007, pp. 796–805.

[18] Yijie Wang, Sijun Li, Research and performance evaluation of data replication technology in distributed storage systems, Comput. Math. Appl. 51 (11) (2006) 1625–1632.

[19] F. Li, K. Yi, J. Jestes, Ranking distributed probabilistic data, in: Proceedings of the International Conference on Management of Data (SIGMOD), ACM, 2009.

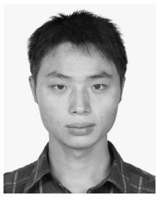
[20] M. Ye, X. Liu, W. Lee, D. Lee, Probabilistic top-*k* query processing in distributed sensor networks, in: Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2010.

[21] L. Deng, F. Wang, B. Huang, Probabilistic threshold join over distributed uncertain data, in: Proceedings of the International Conference on Web-Age Information Management (WAIM), Springer, 2011, pp. 68–80.

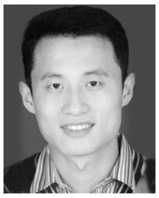
[22] S. Garg, S. Versteeg, R. Buyya, A framework for ranking of cloud computing services, Future Gener. Comput. Syst. 29 (4) (2013) 1012–1023.
[23] S. Wang, B. Ooi, A. Tung, L. Xu, Efficient skyline query processing on peer-to-peer networks, in: Proceedings of the IEEE International Conference on Data Engineering (ICDE), IEEE, 2007, pp. 1126–1135.
[24] A. Vlachou, C. Doulkeridis, Y. Kotidis, M. Vazirgiannis, Skypeer: efficient subspace skyline computation over distributed data, in: Proceedings of the 23rd IEEE International Conference on Data Engineering (ICDE), IEEE, 2007, pp. 416–425.
[25] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, Y. Zhou, Parallel distributed processing of constrained skyline queries by filtering, IEEE Trans. Knowl. Data Eng. 21 (7) (2008) 546–555.
[26] L. Chen, B. Cui, H. Lu, L. Xu, Q. Xu, iSky: efficient and progressive skyline computing in a structured P2P network, in: Proceedings of the 28th IEEE International Conference on Distributed Computing Systems (ICDCS), IEEE, 2008, pp. 160–167.
[27] L. Zhu, Y. Tao, S. Zhou, Distributed skyline retrieval with low bandwidth consumption, IEEE Trans. Knowl. Data Eng. (2009) 384–400.
[28] W. Balke, U. Güntzer, J. Zheng, Efficient distributed skylining for web information systems, in: Proceedings of the International Conference on Extending Database Technology (EDBT), 2004.
[29] S. Sun, Z. Huang, H. Zhong, D. Dai, H. Liu, J. Li, Efficient monitoring of skyline queries over distributed data streams, Knowl. Inf. Syst. 25 (3) (2010) 575–606.
[30] Z. Huang, C. Jensen, H. Lu, B. Ooi, Skyline queries against mobile lightweight devices in MANETs, in: Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE), IEEE, 2006, p. 66.
[31] J. Xin, G. Wang, L. Chen, X. Zhang, Z. Wang, Continuously maintaining sliding window skylines in a sensor network, in: Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA), 2007.
[32] X. Ding, H. Jin, Efficient and progressive algorithms for distributed skyline queries over uncertain data, IEEE Trans. Knowl. Data Eng. 24 (8) (2012) 1448–1462.
[33] X. Wang, Y. Jia, Grid-based probabilistic skyline retrieval on distributed uncertain data, in: Proceedings of the International Conference on Database Systems for Advanced Applications Workshops (DASFAAW), 2011, pp. 538–547.
[34] K. Hose, A. Vlachou, A survey of skyline processing in highly distributed environments, VLDB J. 21 (3) (2012) 359–384.
[35] Y. Wang, X. Li, X. Li, Y. Wang, A survey of queries over uncertain data, Knowl. Inf. Syst. 37 (3) (2013) 485–530.
[36] K. Hose, C. Lemke, K. Sattler, Processing relaxed skylines in PDMS using distributed data summaries, in: Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM), ACM, 2006, pp. 425–434.
[37] E. Wu, Y. Diao, S. Rizvi, High-performance complex event processing over streams, in: Proceedings of the International Conference on Management of Data (SIGMOD), ACM, 2006, pp. 407–418.
[38] A. Vlachou, C. Doulkeridis, Y. Kotidis, M. Vazirgiannis, Efficient routing of subspace skyline queries over highly distributed data, IEEE Trans. Knowl. Data Eng. 22 (12) (2010) 1694–1708.
[39] K. Fotiadou, E. Pitoura, Bitpeer: continuous subspace skyline computation with distributed bitmap indexes, in: Proceedings of the International Workshop on Data Management in Peer-to-Peer Systems (DaMaP), ACM, 2008, pp. 35–42.
[40] S. Wang, Q. Vu, B. Ooi, A. Tung, L. Xu, Skyframe: a framework for skyline query processing in peer-to-peer systems, VLDB J. 18 (1) (2009) 345–362.
[41] J. Rocha-Junior, A. Vlachou, C. Doulkeridis, K. Nørvåg, Agids: a grid-based strategy for distributed skyline query processing, in: Proceedings of the Data Management in Grid and Peer-to-Peer Systems (Globe), 2009.
[42] J. Rocha-Junior, A. Vlachou, C. Doulkeridis, K. Nørvåg, Efficient execution plans for distributed skyline query processing, in: Proceedings of the 14th International Conference on Extending Database Technology (EDBT), ACM, 2011, pp. 271–282.
[43] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, in: Proceedings of the ACM International Conference on the Applications, Technologies, Architectures and Protocols for Computer Communication (SIGCOMM), 2001.
[44] H. Jagadish, B. Ooi, Q. Vu Baton, A balanced tree structure for peer-to-peer networks, in: Proceedings of the International Conference on Very Large Data Bases (VLDB), VLDB Endowment, 2005, pp. 661–672.
[45] Y. Yang, Y. Wang, Towards estimating expected sizes of probabilistic skylines, Sci. China, Ser. F, Inf. Sci. 54 (12) (2011) 2554–2564.
[46] C. Böhm, A. Pryakhin, M. Schubert, The Gauss-tree: efficient object identification in databases of probabilistic feature vectors, in: Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE), 2006.
[47] X. Li, Y. Wang, X. Li, Y. Wang, Parallelizing skyline queries over uncertain data streams with sliding window partitioning and grid index, Knowl. Inf. Syst., http://dx.doi.org/10.1007/s10115-013-0725-8.
[48] A. Sarma, O. Benjelloun, A. Halevy, J. Widom, Working models for uncertain data, in: Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2006.
[49] Y. Tao, X. Xiao, R. Cheng, Range search on multidimensional uncertain data, ACM Trans. Database Syst. 32 (3) (2007) 15–63.
[50] M. Soliman, I. Ilyas, K. Chang, Probabilistic top-*k* and ranking-aggregate queries, ACM Trans. Database Syst. 33 (3) (2008) 1–54.
[51] N. Dalvi, D. Suciu, Efficient query evaluation on probabilistic databases, VLDB J. 16 (4) (2007) 523–544.
[52] T. Ge, S. Zdonik, S. Madden, Top-*k* queries on uncertain data: on score distribution and typical answers, in: Proceedings of the ACM International Conference on Management of Data (SIGMOD), 2009.
[53] X. Li, Y. Wang, X. Li, Y. Wang, Parallel skyline queries over uncertain data streams in cloud computing environments, Int. J. Web Grid Serv. 10 (1) (2014) 24–53.

Xiaoyong Li received the B.S. degree in computer science and technology from the School of Information Science and Technology in Xiamen University, China, in 2006, and received the M.S. and Ph.D. degrees in computer science and technology from the School of Computer Science in the National University of Defense Technology, China, in 2008 and 2013, respectively. He is a research assistant in National University of Defense Technology. He is also a member of CCF, IEEE and ACM. His current research interests lie in the areas of data stream management, network computing, uncertain queries, and cloud computing.

Yijie Wang received the Ph.D. degree in computer science and technology from the School of Computer Science in the National University of Defense Technology, China in 1998. She was a recipient of the National Excellent Doctoral Dissertation (2001), a recipient of Fok Ying Tong Education Foundation Award for Young Teachers (2006) and a recipient of the Natural Science Foundation for Distinguished Young Scholars of Hunan Province (2010). Now she is a Professor in the National Key Laboratory for Parallel and Distributed Processing, National University of Defense Technology. Her research interests include network computing, massive data processing, and parallel and distributed computing.



Xiaoling Li received the B.S. and M.S. degrees in computer science and technology from the School of Computer Science in the National University of Defense Technology, China, in 2007 and 2008, respectively. He is a research assistant in National University of Defense Technology. He is also a student member of CCF and ACM. His current research interests lie in the areas of distributed computing, network virtualization, cloud computing, and trusted computing.



Xiaowei Wang received the B.S., M.S. and Ph.D. degrees in computer science and technology from the School of Computer Science in the National University of Defense Technology, China, in 2003, 2005 and 2012, respectively. His current research interests lie in the areas of massive data management, data mining, and cloud computing.

Jie Yu received the B.S. and M.S. degrees in computer science and technology from the School of Computer Science in the National University of Defense Technology, China, in 2011 and 2013, respectively. He is currently a Ph.D. candidate in the School of Computer Science in the National University of Defense Technology. His current research interests include distributed systems, massive data management and cloud computing.