World Wide Web
DOI 10.1007/s11280-015-0341-5

Tracking frequent items over distributed probabilistic data

Yongxin Tong · Xiaofei Zhang · Lei Chen

Received: 23 October 2014 / Revised: 18 December 2014 / Accepted: 5 March 2015

Abstract Tracking frequent items (also called heavy hitters) is one of the most fundamental queries in real-time data due to its wide applications, such as logistics monitoring, association rule based analysis, etc. Recently, with the growing popularity of Internet of Things (IoT) and pervasive computing, a large amount of real-time data is usually collected from multiple sources in a distributed environment. Unfortunately, data collected from each source is often uncertain due to various factors: imprecise reading, data integration from multiple sources (or versions), transmission errors, etc. In addition, due to network delay and limited by the economic budget associated with large-scale data communication over a distributed network, an essential problem is to track the global frequent items from all distributed uncertain data sites with the minimum communication cost. In this paper, we focus on the problem of *tracking distributed probabilistic frequent items (TDPF)*. Specifically, given k distributed sites $S = \{S_1, \ldots, S_k\}$, each of which is associated with an uncertain database $\mathcal{D}_i$ of size n_i, a centralized server (or called a coordinator) H, a minimum support ratio r, and a probabilistic threshold t, we are required to find a set of items with minimum communication cost, each item X of which satisfies $Pr(sup(X) \geq r \times N) > t$, where $sup(X)$ is a random variable to describe the *support* of X and $N = \sum_{i=1}^{k} n_i$. In order to reduce the communication cost, we propose a local threshold-based deterministic algorithm and a sketch-based sampling approximate algorithm, respectively. The effectiveness and efficiency of the proposed algorithms are verified with extensive experiments on both real and synthetic uncertain datasets.

Y. Tong
State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing, China
e-mail: yxtong@nlsde.buaa.edu.cn

X. Zhang · L. Chen (✉)
Department of Computer Science and Engineering, Hong Kong University of Science Technology, Hong Kong, China
e-mail: leichen@cse.ust.hk

X. Zhang
e-mail: zhangxf@cse.ust.hk

Published online: 05 May 2015

Keywords Frequent itemsets · Distributed · Probabilistic data

1 Introduction

According to a technical report about big data from IBM, 90 % of the data, which is more than 2.5 quintillion bytes, in the world today was created in the last two years alone [48]. Traditional stand-alone data management techniques encounter a bottleneck when deal with this large scale of data, and distributed data management approaches are playing a greater role in the big data era. In addition, with the wide usage of Internet of Things (IoT) and pervasive computing in recent years, a growing number of real-world systems collect data from distributed sites, such as monitoring systems based on sensor networks [28, 30], frameworks of data integration from multiple data crawlers [19], etc. In particular, due to errors generated by hardware measurement and data transmission, the data collected from distributed sites often arises with inherent uncertainty. Thus, we model the data generated from these applications as distributed uncertain data. Let us take a real project, Shipboard Automated Meteorological and Oceanographic System (SAMOS) [35], as example. SAMOS collects a large number of meteorological and near-surface oceanographic observations via research vessels and ships. According to the technical documents of SAMOS, the data collected from SAMOS has the following properties: (1) the data is naturally distributed since the ships are at geographically separated locations; (2) the amount of collected data is often huge. Specifically, the ships in SAMOS continuously generate hundreds of navigational and meteorological records including many parameters (e.g., air temperature, pressure, moisture, rainfall, etc.) in a minute or less; (3) the data is uncertain and unreliable because the data collected in real-time usually includes a lot of noise and imprecise readings. Therefore, the data from SAMOS can be treated as distributed probabilistic data.

Meanwhile, the problem of tracking frequent items (also called heavy hitters) in deterministic real-time data has been widely studied in many real applications [15, 16, 24, 31–33]. The problem of tracking frequent items is to find items whose supports are greater than a specified minimum support threshold. When the concept of frequent items is extended to uncertain environment, existing researches define two different probabilistic semantics: *expected frequent item* [14] and *probabilistic frequent item* [47] as the *support* of an item in uncertain environment becomes a random variable. On one hand, the definition of *expected frequent items* uses the expected support (the expectation of the *support*) to replace the count in deterministic cases. In other words, an item is an *expected frequent item* if the expected support of the item is greater than a specific threshold. On the other hand, the definition of *probabilistic frequent items* employs the random event probability that the *support* is greater than a specific threshold to replace the count in deterministic cases. Namely, an item X is an *probabilistic frequent item* if $Pr(sup(X) \geq r) > t$, where $sup(\cdot)$ is the *support* of X, r is the minimum support ratio, and t is the probabilistic threshold. In the following, we show two real application examples of tracking frequent items over distributed uncertain data.

Example 1 (*A Real-Time Airport Luggage Tracking System*) Figure 1 shows an RFID-based airport luggage monitoring system, which employs multiple RFID readers to monitor the luggages in the airport in real time. Each RFID reader has a broadcast range, which is shown in shadow in Figure 1. Each luggage is attached with a tag and is put on a conveyor belt. A tag can respond to the broadcasting signals from the readers when it enters the broadcast ranges of the corresponding readers. Since there are millions of luggage everyday, it is inefficient to employ only one RFID reader to sense all pieces of luggage on one conveyor

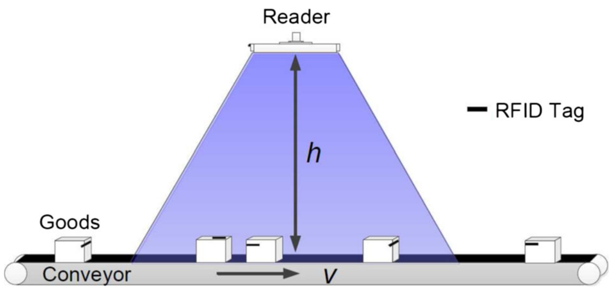


Figure 1 A real-time airport luggage tracking system

belt. Hence, a practical airport luggage monitoring system is a distributed system, which needs to merge real-time information from different RFID readers on multiple conveyor belts.

To optimize the schedule of forklifts, which are used to take the luggage from the conveyor belts to flights, airport managers need to know how many pieces of luggage of each flight are currently on the conveyor belts in real time. If the number of luggage for a flight exceeds a specific threshold, a forklift will be allocated to process the corresponding luggage. Due to the limitation of hardwares and protocols of RFID, i.e., ALOHA [34], the accuracy of response of a tag depends on the distance between the reader and the tag and the signal strength. Therefore, the data collected from RFID readers is usually incomplete and inaccurate. In this example, if the pieces of luggage, which belong to the same flight, are considered as the same item, the above problem of scheduling forklifts is actually equivalent to the problem of finding frequent items in the distributed uncertain monitoring data.

Example 2 (*A Sensors-based Monitoring System*) Another example is about a distributed-sensor-based monitoring system. In this system, we suppose each distributed site has several sensors, which continuously capture and collect readings of different measurements (i.e. density of carbon dioxide, temperature, etc.). All the sites send the collected readings to the coordinator periodically. Due to transmission errors and imprecise measurement, the data collected from multiple sites is usually inconsistent. In other words, though the data is collected at the same site for the same monitored object, contradictory readings may arise. Thus, uncertain data models are more suitable for this type of data [27, 37]. In order to manage the real-time monitoring, a typical tracking goal is to find which monitored values are frequently observed at the global coordinator site.

In summary, the core task of the aforementioned examples is tracking distributed probabilistic frequent items (*TDPF*). For the same reasons and the same motivations of previous studies on tracking distributed deterministic data, the primary goal of *TDPF* is to reduce the communication cost in a system, measured by the total number of communicated messages. For example, in the SAMOS system, cutting down the communication cost would allow more accurate transmission or more diverse measurement. Because of the inherent difference in tracking probabilistic and deterministic data, techniques developed for deterministic scenarios are no longer directly applicable. For example, in an uncertain scenario, we spend at least $\mathcal{O}(n^2)$ time, where n is the size of the uncertain data, to calculate the frequentness probability for an item [47]. On the contrary, the computation cost of calculating the support for an item is only $\mathcal{O}(n)$ in deterministic environment [31]. Therefore, in this paper, we propose solutions to minimize the communication cost and to reduce the computation cost as much as possible in the *TDPF* problem. To summarize, we have made the following contributions:

- We formalize the *tracking distributed probabilistic frequent items (TDPF)* problem and introduce a non-trivial baseline algorithm.
- We design a deterministic algorithm to solve the *TDPF* problem. In particular, two effective pruning methods and an efficient local threshold-based update mechanism are proposed to reduce the communicated bytes, and a theoretical analysis of the total number of communicated messages is provided.
- We develop an efficient and effective sketch-based sampling algorithm to further reduce the communication cost for the worst case.
- Extensive experiments demonstrate the effectiveness and efficiency of the proposed algorithms.

The rest of the paper is organized as follows. The problem definition and a non-trivial baseline method are introduced in Section 2. In Section 3, we present a deterministic algorithm, which includes two pruning methods and a local threshold-based updating mechanism to reduce communicated bytes. Furthermore, we propose a sketch-based sampling algorithm, which not only satisfies the accuracy requirement but also reduces the total number of communicated messages as much as possible. Experimental studies are reported in Section 5. We review existing works in Section 6 and conclude the paper in Section 7.

2 Problem formulation

In this section, we first introduce the distributed system architecture adopted by the tracking problem and some related background concepts, and formally define the *TDPF* problem. Then, a non-trivial baseline algorithm is presented.

2.1 System architecture

We first introduce the system architecture, called the *flat* model, which is widely adopted by many prior works of distributed tracking [13, 22, 23, 25, 27, 45]. In this architecture, there are k distributed sites $S_1, \ldots, S_k$, each of which is associated with an uncertain database $\mathcal{D}_i$ of the size n_i. In addition, there is a centralized server (or called the coordinator), denoted as H, which aims to maintain (an approximation of) query result continuously at all time. Let $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_k$ be the global uncertain database including all the uncertain databases of the k sites and $N = \sum_{i=1}^{k} n_i$ be the total size of $\mathcal{D}$. The system architecture is shown in Figure 2.

Since it is more suitable to model and represent the data collected in real-time in a probabilistic approach, we adopt the *x-relation* model to represent the uncertain data. The *x-relation* model has already been adopted by a lot of previous studies [14, 47] and is suitable for measurement and reading data. In this model, an uncertain database (or data set) consists of a set of tuples, known as *x-tuple*. In our system, each uncertain database $\mathcal{D}_i$ of S_i receives monitored tuples over time. The j-th tuple in $\mathcal{D}_i$ is denoted by $T_{i,j}$. Each tuple contains a set of items. For each item X contained in tuple $T_{i,j}$, i.e. $X \in T_{i,j}$, it is associated with a probability $p_{i,j}(X)$ $(0 \leq p_{i,j}(X) \leq 1)$, which is the possibility that the item X appears in $T_{i,j}$. If $T_{i,j}$ does not contain item X, $p_{i,j}(X) = 0$. Please note the sum of probabilities of items in a tuple is smaller than or equal to 1, i.e. $\sum_{\forall X \in T_{i,j}} p_{i,j}(X) \leq 1$. Furthermore,

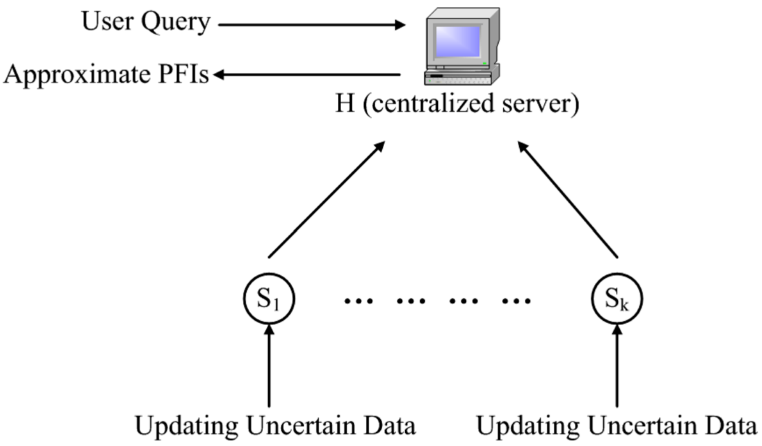


Figure 2 Flat-model-based distributed system architecture

different tuples are assumed to be independent in the *x-relation* model. In fact, an item X in tuple $T_{i,j}$ can be considered as a random variable following the Binomial distribution with probability $p_{i,j}(X)$.

In a deterministic database, the number (or the so-called count) of tuples containing an item X is called the *support* of X [2, 3]. However, in the *x-relation* model, the *support* of an item becomes a random variable. In our system, we first denote $sup_i(X)$ as the *support* of item X in $\mathcal{D}_i$. In other words, $sup_i(X)$ is the sum of n_i random variables following different Binomial distributions, each with probability $p_{i,j}(X)$. Then globally, for all the tuples in $\mathcal{D}$, the *support* of an item X, denoted by $sup(X)$, is actually the sum of $\sum_{i=1}^{k} n_i$ random variables following different Binomial distributions. Please note that $sup(X)$ is the sum of $\sum_{i=1}^{k} n_i$ bounded random variables since $\forall p_{i,j}(X) \in [0, 1]$.

2.2 Problem statement

In this subsection, we first define some important concepts, and then formulate the problem of tracking probabilistic frequent items in distributed systems.

Definition 1 (Expected Support) Given an *x-relation* uncertain database $\mathcal{D}$ that includes N tuples, and an item X, the expected support of X is the mean of $sup(X)$,

$$E(sup(X)) = \sum_{i=1}^{N} p_w(X) \tag{1}$$

where $p_w(X)$ is the probability that the item X appears in the w-th tuple. Please note that $sup(X)$ is a random variable rather than a simple count in uncertain environment.

Definition 2 (Frequentness Probability) Given an *x-relation* uncertain database $\mathcal{D}$ that includes N tuples, a minimum support ratio r ($0 \le r \le 1$), and an item X, the frequentness probability of X is shown as follows:

$$Pr\{sup(X) \ge r \times N\} = \sum_{i=r\times N}^{N} Pr\{sup(X) = i\} \tag{2}$$

Definition 3 (Exact Probabilistic Frequent Item) Given an *x-relation* uncertain database $\mathcal{D}$ that includes N tuples, a minimum support ratio r ($0 \leq r \leq 1$), and a probabilistic threshold t, an item X is a probabilistic frequent item in $\mathcal{D}$ if the frequentness probability of X is greater than t, namely,

$$Pr\{sup(X) \geq r \times N\} > t \tag{3}$$

Although Definition 3 describes the frequentness probability of an item precisely, it is computationally costly. According to existing studies [5], the cost of calculating frequentness probability of an item is at least $\mathcal{O}(N^2)$, where N is the size of the uncertain database $\mathcal{D}$. Hence, it is infeasible for real-time queries to compute the exact frequentness probability for each item. As claimed in some previous studies [14, 25, 45], exact solution is not required in practice, and the definition of approximate probabilistic frequent item is usually adopted instead.

Definition 4 (Approximate Probabilistic Frequent Item (APFI)) Given an *x-relation* uncertain database $\mathcal{D}$ that includes N tuples, a minimum support ratio r ($0 \leq r \leq 1$), a probabilistic threshold t, a minimum support ratio error ϵ, and a probabilistic threshold error θ, an item X is an approximate probabilistic frequent item in $\mathcal{D}$ if

$$\begin{aligned} &1) Pr\{sup(X) \geq r \times N\} > t \\ &2) X\ \textit{cannot satisfy}\ Pr\{sup(X) \geq (r - \epsilon)N\} < (1 - \theta)t \end{aligned} \tag{4}$$

According to the aforementioned system architecture, we formulate the problem of tracking distributed probabilistic frequent items (*TDPF*) as follows.

Definition 5 (Tracking Distributed Probabilistic Frequent Items (*TDPF*)) Given k distributed sites $S = \{S_1, \ldots, S_k\}$, each of which is associated with an *X-relation* uncertain database $\mathcal{D}_i$ of size n_i, a centralized server H, a minimum support ratio r, and a probabilistic threshold, t, the *TDPF* problem is to find a set of APFIs with the minimum communication cost.

Table 1 summarizes the symbols. We use the following example to illustrate the definitions above.

Example 3 Given a distributed uncertain database in Table 2 with two sites, each of which receives two tuples in time slots $Time_1$ and $Time_2$ respectively, $r = 0.6$, and $t = 0.4$, the frequentness probabilities of $\{A\}$ in S_1 and S_2 are 0.64 and 0.25 respectively. Thus, $\{A\}$ is a probabilistic frequent item in S_1 but not in S_2. However, $\{A\}$ is probabilistic frequent item in the whole database, whose frequentness probability is 0.4, since the central server, H, considers the frequentness probability of $\{A\}$ globally.

2.3 Baseline algorithm

Similar to previous work of querying in distributed probabilistic databases [27, 37], we assume that the probability distribution of each item at an arbitrary site is already stored, and we try to find which items are (approximately) probabilistic frequent in the global centralized server according to the local probability distributions. A naive solution is to upload all the data in the distributed sites to the centralized server and then find the frequent items. However, the communication cost of such naive solution is prohibited for large-scale

Table 1 Summary of notations

Notation	Meaning
S_i	an *x-relation model*-based uncertain local site
$\mathcal{D}_i$	an *x-relation model*-based uncertain database corresponding to S_i
$\mathcal{D}$	the global uncertain database, $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_k$
n_i	the size of D_i
H	a centralized server (or called coordinator)
N	$\sum_{i=1}^{k} n_i$
$p_{i,j}(X)$	the possibility that the item X appears in the j-th tuple $T_{i,j}$ in $\mathcal{D}_i$
r	the specific minimum support ratio
t	the specific probabilistic threshold
ϵ	the error of minimum frequent ratio
θ	the error of probabilistic threshold
$sup_i(X)$	the *support* of an item X in $\mathcal{D}_i$
$sup(X)$	the *support* of an item X in the global $\mathcal{D}$
$E_i(sup(X))$	the expected support of an item X in S_i
$E(sup(X))$	the global expected support of an item X

networks. Therefore, the major challenge is to calculate the global frequentness probability for each item with minimum communication cost. In this subsection, we give a non-trivial baseline solution.

As shown in Algorithm 1, the baseline algorithm first adopts a divide-and-conquer framework, called the merging algorithm (Algorithm 2), to merge the local probability distributions of each item. It divides k sites into two groups: $U_1 = \{S_1, \ldots, S_{\frac{k}{2}}\}$ and $U_2 = \{S_{\llcorner \frac{k}{2} \lrcorner +1}, \ldots, S_k\}$, and recursively repeats the divide process until each group includes only one site. Then, it merges these probability distributions in the conquer phase based on a convolution computation. The computation of convolution is shown in the following formula. Finally, the complete probability distribution of *support* is obtained when the algorithm terminates.

$$PD_X[k] = \sum_{i=0}^{k} PD_X^1[i] \times PD_X^2[k-i] \tag{5}$$

where PD_X^1 and PD_X^2 are used to store the probability distributions of $sup(X)$ in U_1 and U_2, respectively.

Hence, the computational complexity of the merging algorithm for an item X is $\mathcal{O}(l^k)$, where l is the number of different values of $sup(X)$ in the global probability distribution of $sup(X)$. In order to enhance its efficiency, we can utilize the Fast Fourier Transform (FFT) technique to speed up the merging algorithm, and the computational complexity of the

Table 2 A distributed uncertain database

Time	Sites	
	S_1	S_2
$Time_1$	{A (0.8), B(0.2)}	{A (0.8), C(0.2)}
$Time_2$	{A (0.5), C(0.5)}	{A (0.5), D(0.5)}

accelerated merging algorithm becomes $\mathcal{O}(l^{\frac{k}{2}} logl^{\frac{k}{2}})$. If the central server already receives m distinct items in the past t unit time, the total worst computational complexity of the baseline algorithm is $\mathcal{O}(m \times t \times l^{\frac{k}{2}} logl^{\frac{k}{2}})$. The framework of the non-trivial baseline algorithm is shown in Algorithm 1.

```
Algorithm 1: Baseline Algorithm
1 while A tuple $T$ arrives at any uncertain database $\mathcal{D}_{\rangle}$ in the local site $S_i$ do
2     for each item $X$ which is included in $T$ do
3         call the Merging Algorithm to calculate the probability distribution for the
          item $X$
4 return the set of probabilistic frequent items;
```

```
Algorithm 2: Merging Algorithm
   Input: an uncertain dataset $\mathcal{D}$, an item $X$
   Output: The probability distribution of $sup(X)$, $PD_X$
1  $N \leftarrow |\mathcal{D}|$;
2  if $N \neq 1$ then
3      Split $\mathcal{D}$ into two groups $U_1$ and $U_2$;
4      $PD_X^1 \leftarrow DC(U_1, X)$;
5      $PD_X^2 \leftarrow DC(U_2, X)$;
6      $PD_X \leftarrow$ FFT-based Convolution$(PD_X^1, PD_X^2)$;
7      return $PD_X$;
8  else
9      $PD_X[0] \leftarrow 1 - p_1$;
10     $PD_X[1] \leftarrow p_1$;
11     return $PD_X$;
```

3 An improved deterministic approach

As discussed in the baseline algorithm, a local site sends the new probability distributions of some items to the coordinator when the items are updated. Thus, too much redundant information is updated. Since saving communication cost is our primary goal, we analyze the following two cases that may consume unnecessary communication cost. (1)*What data should be updated?* In the baseline algorithm, each local site always sends the new probability distributions. In fact, there are usually many infrequent items. If we can prune them, then their probability distributions do not need to be uploaded. (2)*When the updated data should be sent?* In the baseline algorithm, the local site sends the new update as long as a new tuple arrives. Unfortunately, new tuple arrives frequently, so the baseline algorithm induces much unnecessary communication cost. Thus, an effective update mechanism is needed to minimize the number of updates while the correctness of results is guaranteed.

Based on the aforementioned analysis, we propose a framework to reduce the update cost. In our framework, we first conduct a bounding analysis, which identifies the cases where update of probability distributions is needed. Specifically, we calculate the upper and lower bounds for the frequentness probability of an item using the expected support of this item only. With these two bounds, we can not only filter out infrequent items as many as possible, but also identify frequent items, whose update of probability distributions is not needed. Then, based on the bounding methods, we propose an effective threshold-based update mechanism in Section 3.2, which predetermines a set of thresholds of expected supports for each local site and uses these thresholds to check the frequentness probability of each item in the coordinator. In particular, we design two different strategies to calculate the set of thresholds and discuss their worst-case communication costs. The framework of the deterministic algorithm is shown in Algorithm 3.

Algorithm 3: A Deterministic Algorithm Framework

```
1  For Local Sites:
2  if an item is locally renewed, and the threshold-based update
   mechanism(Section 3.2) allows to send then
3      Send the expected support of this item to the coordinator;
4  if the coordinator requests the probability distribution of support of an item in this
   local site then
5      Send the probability distribution of the support of the specified item;
6  For Coordinator:
7  if the expected support of an item are received then
8      if The bounding or pruning techniques (Section 3.1) cannot determine whether
       this item is APFI then
9          Request each local site to upload the local probability distribution of this
           item;
10     else
11         Decide this item is APFI or not globally;
```

3.1 Early bounding and pruning

Due to the high communication cost, it is infeasible to set up the communication between the coordinator and all sites once any new update in one site arrives. Furthermore, some items may be always infrequent globally, thus the communication induced by infrequent items is redundant. In this subsection, we propose tight lower and upper bounds on the frequentness probability. The two bounds only use the expected support of an item without using the complete probability distribution. Then, based on the bounds, we design an early pruning strategy using the expected support only.

We first introduce the lower and upper bounds of frequentness probability in the following lemma.

Lemma 1 *(Lower and Upper Bounds) Given k distributed sites $S = \{S_1, \ldots, S_k\}$, each of which is associated with an uncertain database $\mathcal{D}_i$ of size n_i, a centralized server H, a minimum support ratio r, a minimum support ratio error ϵ, and an item X, the upper bound and the lower bound of the frequentness probability of X in H are shown as follows,*

$$\begin{cases} Pr\{sup(X) \geq (r-\epsilon)N\} \geq 1 - e^{-\frac{\mu(1-(rN/\mu)^2)}{2}} & \mu \geq (r-\epsilon)N \\ Pr\{sup(X) \geq (r-\epsilon)N\} < min\{1, (\frac{e^{\lambda}}{(1+\lambda)^{1+\lambda}})^{\mu}\} & \mu < (r-\epsilon)N \end{cases} \quad (6)$$

where $\mu = E(sup(X)) = \sum_{i=1}^{k} E_i(sup(X))$, in which $E_i(sup(X)) = \sum_{j=1}^{n_i} p_{i,j}$ is the expected support of the item X at site S_i, $N = \sum_{i=1}^{k} n_i$, and $\lambda = \frac{n(r-\epsilon)^2}{r\mu} - 1$.

Proof According to the definition of *support* of an item X over $\mathcal{D}$ in Section 2, $sup(X)$ is actually the sum of $\sum_{i=1}^{k} n_i$ bounded random variables following Binomial distributions. The global expected support of an item X in $\mathcal{D}$, denoted by $E(sup(X))$, is $\sum_{i=1}^{k}\sum_{j=1}^{n_i} p_{i,j}$. Thus, for each item, its support satisfies the condition of Chernoff inequality [9]. According to the Chernoff inequality, we know that

$$Pr\{sup(X) \geq (1-\alpha)\mu\} \geq 1 - e^{-\frac{\mu\alpha^2}{2}} \ (0 < \alpha < 1)$$

when $\mu \geq (r-\epsilon)N$. Let $(1-\alpha)\mu = (r-\epsilon)N$. We know $\alpha = 1 - \frac{(r-\epsilon)N}{\mu}$. Replacing α in the aforementioned formula,

$$\begin{aligned} &Pr\{sup(X) \geq (r-\epsilon)N\} \\ &\geq 1 - e^{-\frac{\mu(1-((r-\epsilon)N/\mu)^2)}{2}} \\ &> 1 - e^{-\frac{\mu(1-(rN/\mu)^2)}{2}} \end{aligned}$$

Thus, the lower bound of frequentness probability holds.

When $\mu < (r-\epsilon)N$,

$$Pr\{sup(X) \geq (1+\lambda)\mu\} < \left(\frac{e^{\lambda}}{(1+\lambda)^{(1+\lambda)}}\right)^{\mu}$$

Let $(1+\lambda)\mu = (r-\epsilon)N$, we know $\lambda = \frac{(r-\epsilon)N}{\mu} - 1$ and,

$$Pr\{sup(X) \geq (r-\epsilon)N\} < \left(\frac{e^{\lambda}}{(1+\lambda)^{1+\lambda}}\right)^{\mu}$$

Since the $Pr\{sup(X) \geq (r-\epsilon)N\} \leq 1$, we can obtain the above upper bound. Therefore, the lemma holds. □

Complexity of the Bounding Since the aforementioned lower and upper bounds adopt the expected support to approximate the frequentness probability, we only need to sum the expected support of each item over all local sites. Thus, the time complexity and communication cost in the coordinator are $\mathcal{O}(k)$, where k is the number of sites.

According to Definition 4, we can define a pair of pruning rules based on the minimum frequent ratio r and the minimum frequent ratio error $r - \epsilon$. Based on the upper and lower bounds, we can filter out infrequent items early and identify some probabilistic frequent items by Lemma 2.

Lemma 2 *(Early Bounding and Pruning)*

Given k distributed sites $S = \{S_1, \ldots, S_k\}$, each of which is associated with an uncertain database $\mathcal{D}_i$ of size n_i, a centralized server H, a minimum support ratio r, a probabilistic threshold t, a minimum support ratio error ϵ, a probabilistic threshold error θ, and an item X, we can obtain the following two early bounding or pruning.

(1) *X can be safely pruned if the upper bound is smaller than $(1-\theta)t$;*
(2) *X must be an $APFI$ if the lower bound is greater than t.*

Proof According to Definition 4, an item X is an approximate probabilistic frequent item ($APFI$) in $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_k$ if and only if the following two conditions are satisfied. On one hand, $Pr\{sup(X) \geq (r - \epsilon)N\} < (1 - \theta)t$ must not hold if X is an $APFI$. According to the upper bound in Lemma 1, X must not be an $APFI$ if the upper bound is smaller than $(1 - \theta)t$. On the other hand, $Pr\{sup(X) \geq r \times N\} > t$ needs to be satisfied if X is an $APFI$. Based on the lower bound in Lemma 1, X must be $APFI$ if the lower bound is greater than t. Thus, Lemma 2 holds. □

In summary, we can determine whether an item is approximately probabilistic frequent early using the expected support of this item based on Lemma 2. Thus, for the bounded items, we can design an effective update mechanism by updating the expected supports only. For each of the remaining items, which cannot be bounded, the coordinator has to notify all the local sites, and then the local sites will upload the probability distribution of *support* of this item. According to the early bounding and pruning method, we propose an effective expected support threshold-based update mechanism in the next subsection.

3.2 Local threshold-based update mechanisms

In this subsection, we first introduce the basic idea of our local threshold-based update mechanism, which predetermines a series of thresholds of expected supports in each local site and uses these thresholds to estimate the frequentness probability for each item in the coordinator. Then, we further design two different strategies and discuss their worst-case communication costs.

3.2.1 Basic Idea

In this part, we introduce the basic idea of the local threshold-based update mechanism, which is shown as follows. Each local site sets a series of thresholds of the expected support. When the expected support of an item exceeds each threshold, the local site sends the updated expected support to the coordinator. After receiving k updates of an item, the coordinator adopts the early bounding and pruning methods to test the item. If the item is not bounded, the coordinator requires all the local sites to upload the probability distributions of this item and further verifies its frequentness. The update mechanism repeats the above process to track the set of $APFIs$. In the following, we define some notations of our update mechanism.

Assume that the i-th local site, S_i, maintains a series of thresholds, $t_{i,j}$ $(j = 0, 1, \ldots)$ and $t_{i,j} < t_{i,j+1}$, where $t_{i,j}$ is the j-th threshold in the local site S_i. Let $E_i(sup(X))$ be the current expected support of the item X in S_i. Suppose $t_{i,j}$ is the current local threshold that is just exceeded by $E_i(sup(X))$, and $t_{i,j} \leq E_i(sup(X)) \leq t_{i,j+1}$ is satisfied. As long as the new $E_i(sup(X)) > t_{i,j+1}$, the local site sends $E_i(sup(X))$ to the coordinator and resets the new local threshold to $t_{i,j+1}$. In other words, before the coordinator receives the new $E_i(sup(X))$, the $E_i(sup(X))$ must lie between the two thresholds, $t_{i,j}$ and $t_{i,j+1}$. Hence, the maximum error of the expected support $E_i(sup(X))$ in S_i is given by $t_{i,j+1} - t_{i,j}$, where $t_{i,j}$ is the current local threshold. Since each pair of adjacent thresholds, $t_{i,j}$ and $t_{i,j+1}$, must satisfy the aforementioned requirement, the total number of updates sent from the local sites to the coordinator is equivalent to the number of thresholds at the local sites. In order to minimize the communication cost, how to predetermine and assign a series of thresholds is the core challenge, which is formally stated as follows.

Definition 6 (Threshold Assignment Problem) Given k distributed sites $S = \{S_1, \ldots, S_k\}$, a coordinator H, and the minimum support ratio error ϵ, each determined threshold $t_{i,j}$ in S_i needs to satisfy the following constraints,

- $t_{i,j+1} > t_{i,j}$ and $t_{i,0} = 0$;
- $\sum_{i=1}^{k}(t_{i,j+1} - t_{i,j}) \leq \epsilon \sum_{i=1}^{k} E_i(sup(X))$;

In the aforementioned definition, the first constraint means that the thresholds are monotonically increasing as j increases. This ensures that the error between two thresholds, $t_{i,j+1} - t_{i,j}$, is never negative. The second constraint guarantees that the maximum total error in the coordinator satisfies the error requirement in the definition of approximate probabilistic frequent items.

In the remaining parts of this subsection, we present two threshold assignment mechanisms to determine the local thresholds at each local site and discuss their communication costs, respectively.

3.2.2 *A simple threshold assignment mechanism*

In this part, we present a simple threshold assignment mechanism. According to Definition 4, we know that the maximum global error is ϵN, where N is the size of the global uncertain database $\mathcal{D}$ at coordinator H. In order to control the maximum total error of each item in the coordinator, a simple solution is to set the j-th threshold at S_i to $t_{i,j} = \frac{jr\epsilon N}{k}$. In fact, this simple method uniformly divides and transfers the maximum global error into each local site, which is called the *simple threshold assignment approach*. Based on the setting, we have total error $\sum_{i=1}^{k}(t_{i,j+1} - t_{i,j}) \leq r\epsilon N < \epsilon N$. The total error satisfies the definition of approximate probabilistic frequent items, and the maximum number of updates of each item is $\frac{\epsilon N}{\frac{jr\epsilon N}{k}} = \frac{k}{r\epsilon}$. To sum up, based on the simple threshold assignment mechanism, we can obtain the following lemma.

Lemma 3 *(Communication Cost of the Simple Threshold Assignment) The total cost of communication between all the sites and the coordinator using the simple threshold assignment approach is $\mathcal{O}\left(\frac{vk}{r\epsilon}\right)$, where v is the number of distinct items in the coordinator.*

Proof According to the idea of the simple threshold assignment mechanism, this mechanism uniformly splits and transfers the maximum global error ϵn into each local site. For each item, the maximum number of updates from all local sites to the coordinator is $\frac{k}{r\epsilon}$. Thus, the total communication cost of this mechanism for all items is $\mathcal{O}\left(\frac{vk}{r\epsilon}\right)$, where v is the number of distinct items in the coordinator. □

3.2.3 *An improved threshold assignment mechanism*

As discussed in the last part, the simple threshold assignment approach uniformly splits the maximum global error. However, in many real-world cases, it is unrealistic that the errors from different local sites are similar. Furthermore, with the increasing size of the collected data, the simple threshold assignment approach does not scale up well to large datasets. For this reason, we propose an improved threshold assignment mechanism, which is to assign thresholds according to the proportion of the size of the data. A formal definition of the improved threshold assignment mechanism is shown as follows.

Let the $(j+1)$-th threshold at S_i is $t_{i,j+1} = (1+\epsilon)t_{i,j}$ and $t_{i,0} = 0, t_{i,1} = 1$. At each local site, the maximum error is $t_{i,j+1} - t_{i,j} = \epsilon t_{i,j}$. Therefore, the maximum global error in the coordinator is,

$$\sum_{i=1}^{k}(t_{i,j+1} - t_{i,j}) = \sum_{i=1}^{k}\epsilon t_{i,j} \leq \epsilon\sum_{i=1}^{k}E_i(sup(X)) < \epsilon N \tag{7}$$

Thus, for each item, this threshold assignment mechanism satisfies the error requirement in the definition of approximate probabilistic frequent item. Hence, we have the following lemma.

Lemma 4 *(Communication Cost of the Improved Threshold Assignment) The total cost of communication between all the sites and the coordinator using the improved threshold assignment approach is $\mathcal{O}\left(\frac{vk}{\epsilon}log\frac{N}{k}\right)$, where v is the number of distinct items in the coordinator.*

Proof For an item X, let $E_i(sup(X))$ and $t_{i,j}$ be the current expected support of X and the current local threshold in S_i respectively, we have $t_{i,j} \leq E_i(sup(X)) < t_{i,j+1}$. According to the improved threshold assignment approach, $t_{i,j} = (1+\epsilon)^{j-1}$. Thus, the number of updates for item X at S_i is

$$j = 1 + log_{1+\epsilon}t_{i,j} \leq 1 + \frac{logE_i(sup(X))}{log(1+\epsilon)} \tag{8}$$

Thus, the global communication cost for item X is

$$\sum_{i=1}^{k} j \leq k + \sum_{i=1}^{k}\frac{logE_i(sup(X))}{log(1+\epsilon)} < \frac{k}{\epsilon}log\frac{N}{k} \tag{9}$$

Therefore, the total communication cost of this mechanism for all items is $\mathcal{O}\left(\frac{vk}{\epsilon}log\frac{N}{k}\right)$, where v is the number of distinct items in the coordinator. $\square$

3.3 A threshold-based deterministic algorithm

In this subsection, we propose the complete deterministic algorithm, which incorporates the aforementioned local threshold-based update mechanism to reduce the communication cost as much as possible. Furthermore, the communication cost in the worst case is also given. The pseudo code of this algorithm is shown in Algorithm 4.

Algorithm 4: The Complete Deterministic Algorithm

```
Input: $k$ sites: $S_1, \ldots, S_k$, a coordinator $H$, a minimum support ratio $r$, a
       probabilistic threshold $t$, a minimum support ratio error $\epsilon$, and a
       probabilistic threshold error $\theta$
Output: A set of approximate probabilistic frequent items $APFI$
1  initialize a set of thresholds based on the improved threshold assignment mechanism;
2  For the $i$-th Local Site:
3  while a new tuple $T$ arrives do
4      for $X \leftarrow 1$ to $v$ do
5          $j \leftarrow 0$;
6          if $j \in X$ then
7              $S_i$ maintains the probability distribution and the expected support of $j$;
8          if $E_i(sup(X)) > t_{i,j}$ then
9              $S_i$ sends $E_i(sup(X))$ and $n_i$ to $H$;
10             $j \leftarrow j + 1$;
11 if the coordinator $H$ requests the probability distribution of the item $X$ then
12     $S_i$ sends the probability distribution of the item $X$;
13 For Coordinator:
14 while $H$ receives $E_i(sup(X))$ and $n_i$ from $S_i$ do
15     $E(sup(X)) \leftarrow E(sup(X)) + E_i(sup(X))$;
16     $N \leftarrow N + n_i$;
17     if $X$ can be bounded as an $APFI$ then
18         $APFI \leftarrow APFI \cup X$ (Lemma 2);
19     else if $X$ cannot be bounded or pruned then
20         $H$ requests the probability distribution of $X$ to $k$ sites;
21 while $H$ receives the probability distributions of $X$ from $k$ sites do
22     $H$ decides whether $X$ is $APFI$;
23 return $APFI$;
```

In Algorithm 4, we initialize the set of thresholds by the improved threshold assignment mechanism in line 1. Then, the algorithm can be divided into two parts: the local site part and the coordinator part. Lines 1-12 describe the operations at each local site. When a tuple T is received by the i-th site S_i, S_i first maintains the probability distributions and the expected supports of the items, which are contained in T in line 7. For each item, S_i sends its expected support to H if its expected support reaches a new threshold. Furthermore, S_i uploads the probability distribution of an item if the coordinator requests in lines 11-12. Lines 13-23 are about the coordinator. When the coordinator receives the updated information in lines 13-20, it updates the current global size and the corresponding expected support. Then, the coordinator tests whether this item can be filtered out or be bounded according to Lemma 2. If the item cannot be bounded, the coordinator has to request all the probability distributions of this item from the k local sites and calculate the frequentness probability to further verify it frequentness in lines 21-22.

The communication cost of this algorithm is given in the following theorem.

Theorem 1 *(Communication Cost of the Deterministic Algorithms) The total cost of communication between all the sites and the coordinator using the deterministic algorithm is $\mathcal{O}(\frac{v_1 k}{\epsilon} log \frac{N}{k} + v_2 N)$, where v_1 and v_2 are respectively the numbers of items which can be bounded and those cannot be bounded according to Lemma 2.*

Proof Let $v = v_1 + v_2$. Based on Lemma 4, the communication cost of v_1 items is $\mathcal{O}(\frac{v_1 k}{\epsilon} log \frac{N}{k})$. For the v_2 items, this algorithm has to collect the complete probability distributions for them. Hence, the communication cost of v_2 is $\mathcal{O}(v_2 N)$, where N is the size of $\mathcal{D}$ at H, namely the total number of global tuples. □

To explain the deterministic algorithm, we illustrate it via the following example.

Example 4 (*The Deterministic Algorithm*) Given a distributed uncertain database as shown in Table 2, $r = 0.6$, $t = 0.5$, $\epsilon = \theta = 0.1$, $\delta = 0.01$, the set of thresholds at site S_1 is initialized as follows: $t_{1,0} = 0, t_{1,1} = 1, t_{1,2} = 1.1, t_{1,3} = 1.21, \ldots$. After S_1 receives the first tuple, $E_1(sup(A)) = 0.8$. Since the current local threshold for the item A is $t_{1,0} = 0$, S_1 sends $E_1(sup(A)) = 0.8$ and $n_1 = 1$ to H. Meanwhile, the current local threshold for A is changed to t_1. Furthermore, the coordinator can return A as an $APFI$ if the coordinator only receives the $E_1(sup(A)) = 0.8$ in the global system. If the coordinator receives $E_1(sup(A)) = 1.6$, $E_2(sup(A)) = 0.9$, $n_1 = n_2 = 2$, the item A cannot be bounded or pruned since $E(sup(A)) = E_1(sup(A)) + E_2(sup(A)) = 2.5$ and $N = n_1 + n_2 = 4$ according to Lemma 2.

4 A sketch-based sampling approach

Even though the proposed deterministic algorithm intends to track approximate probabilistic frequent items by updating the expected support as less often as possible, this algorithm still has to transmit the complete probability distributions of the items, which cannot be pruned or bounded by Lemma 2. In this section, we design a sketch-based sampling algorithm, which reduces the communication cost in the worst case. The basic idea of the sampling algorithm is to use sketches to reduce communication and computation cost. In the following, we first briefly introduce our main idea in Section 4.1, and then describe the sampling algorithm in Section 4.2, and finally prove the correctness of the sampling algorithm in Section 4.3.

4.1 Main ideas

Our main idea is derived from the *AMS Sketch*[4]. We first review the basic idea of *AMS Sketch*, which is widely used to estimate frequency moments. The *AMS Sketch* consists of a matrix, each element of which stores a sampled count. For the sketch, it first calculates two important parameters, the number of rows and that of columns, based on user-specific approximation errors. Then this sketch performs sampling for each element of the matrix and maintains the sampled counts. After sampling, an *AMS Sketch-based* algorithm is usually adopted to take the average of each row in the matrix and use the median of the averages from all the rows as the final estimator. It aims to reduce the influence of variances in the sampling by taking the average of each row. Meanwhile, it also enhances the probability of success by using the median among the averages of all distinct rows.

Inspired by the idea of *AMS Sketch*, we first construct an *AMS Sketch* for each item at each local site and the coordinator. For each item, our algorithm maintains an exact sampling result at each local site and performs an effective sampling method to decide when the local results should be updated to the coordinator. Finally, when the coordinator tries to find the current $APFIs$, our algorithm uses the *average-median* approach to calculate the estimated approximate frequentness probability for each item and decide which items are $APFIs$.

4.2 Algorithm description

The pseudo code of our algorithm is shown in Algorithm 5. First, we initialize two parameters, m_1 and m_2, which are the number of rows and columns in the sketch matrix, respectively. In line 1, let $m_1 = 2ln(\frac{1}{\delta})$, where δ is a parameter about the confidence error of this sampling algorithm, and $m_2 = \frac{8}{\theta^2 t}$, where t and θ are the probabilistic threshold and the error of the probabilistic threshold, respectively. Then, we maintain $m_1 \times m_2$ sampled counts for each item in the coordinator and each local site in line 2. Given the item X, one of the $m_1 \times m_2$ sampled counts is denoted as $C_{i,j}^{X,l}$ at the l-th local site (or $C_{i,j}^{X}$ in the coordinator), where $1 \le i \le m_1$ and $1 \le j \le m_2$. Our sampling algorithm consists of two parts, the local site part in lines 3-10 and the coordinator part in lines 11-25, respectively.

Algorithm 5: Sketch-based Sampling Algorithm

```
   Input: k sites: $S_1, \ldots, S_k$, a coordinator $H$, a minimum support ratio $r$, a error of
          minimum support ratio $\epsilon$, a probabilistic threshold, $t$, a error of probabilistic
          threshold, $\theta$
   Output: A set of approximate probabilistic frequent items $APFI$
1  $m_1 \leftarrow 2ln(\frac{1}{\delta})$, $m_2 \leftarrow \frac{8}{\theta^2 t}$;
2  initialize $m_1 \times m_2$ counters with zero for each item;
3  For the $l$-th Local Site:
4  while a new tuple $T$ arrives do
5      for $i \leftarrow 1$ to $m_1$ do
6          for $j \leftarrow 1$ to $m_2$ do
7              if The item $X$ is sampled from $T$ then
8                  $C_{i,j}^{X,l} \leftarrow C_{i,j}^{X,l} + 1$;
9                  if $C_{i,j}^{X,l}$ can be uploaded with the probability $p$ then
10                     send $C_{i,j}^{X,l}$ to $H$;
11                     $C_{i,j}^{X,l} \leftarrow 0$;
12 For the Coordinator:
13 while $H$ receives the update of $C_{i,j}^{X,l}$ from $S_l$ do
14     $C_{i,j}^{X} \leftarrow C_{i,j}^{X} + C_{i,j}^{X,l}$;
15 for each item $X$ do
16     for $i \leftarrow 1$ to $m_1$ do
17         for $j \leftarrow 1$ to $m_2$ do
18             if $C_{i,j}^{X} > t$ then
19                 $\zeta_{i,j}^{X} \leftarrow 1$;
20             else
21                 $\zeta_{i,j}^{X} \leftarrow 0$;
22             $\zeta_{i}^{X} \leftarrow \zeta_{i}^{X} + \zeta_{i,j}^{X}$;
23         $Y_i^X \leftarrow \frac{\zeta_i^X}{m_2}$;
24     $Y^X \leftarrow median\{Y_1^X, \ldots, Y_{m_1}^X\}$;
25     if $Y^X > (1-\theta)t$ then
26         $APFI \leftarrow APFI \cup X$;
27 return $APFI$;
```

At a local site S_l, when a new tuple arrives, we perform $m_1 \times m_2$ units of sampling operation, each of which includes two random components, in lines 5-10. We first decide which item appears in this tuple. Please note that there exists at most one sampled item in each tuple since different items in a tuple are mutually exclusive according to the *x-relation model*. Once an item X is sampled as the outcome, we add 1 to $C_{i,j}^{X,l}$ and decide whether $C_{i,j}^{X,l}$ should be uploaded with probability p in lines 7-11. If so, the count of $C_{i,j}^{X,l}$ at this local site is sent to the coordinator, and is then reset to zero (in lines 10-11). Otherwise, $C_{i,j}^{X,l}$ remains unchanged.

On the other hand, the coordinator H keeps receiving the uploaded $C_{i,j}^{X,l}$ from different local sites and accumulates them to $C_{i,j}^{X}$ in lines 13-14. When a *TDPF* query is requested, the coordinator determines whether each $C_{i,j}^{X}$ is greater than $r \times N$, where r is the minimum support ratio (in lines 15-23). If so, $\zeta_{i,j}^{X} = 1$, and otherwise $\zeta_{i,j}^{X} = 0$ (in lines 18-21). Then, based on the *AMS Sketch*, let $Y_i^X = \frac{\sum_{j=1}^{s_2} \zeta_{i,j}^X}{sm_2}$ in line 22. For m_1 Y_i^X $(1 \le i \le m_1)$, let $Y^X = median\{Y_1^X, \ldots, Y_{m_1}^X\}$ in line 24. In lines 25-26 our algorithm decides whether X is an $APFI$ or not according to the following conditions,

$$\begin{cases} X \, is \, APFI & Y^X > t \\ X \, is \, not \, APFI & Y^X < (1-\theta)t \end{cases} \tag{10}$$

To sum up, in our algorithm, the coordinator first collects $m_1 \times m_2$ samples of the support of each item from k local sites. In this process, each local site first maintains $m_1 \times m_2$ local sampled counts for each item and then randomly decides whether to upload a count when it is increased. When a *TDPF* query is required, for each item, the coordinator tests which of the $m_1 \times m_2$ counts are greater than $r \times n$ and obtains $m_1 \times m_2$ $\zeta_{i,j}^X$. The matrix $\zeta_{i,j}^X$ is utilized to decrease the variance of samples via taking the average of $\zeta_{i,j}^X$ in each row and to improve the success probability of the sampling algorithm by taking the median of the m_1 averages. According to our algorithm, the probability that an item is correctly decided is at least $1 - \delta$, which will be proved later. To explain our sampling algorithm, we illustrate it via the following example.

Example 5 (*The AMS-Sketch-based Sampling Algorithm*) Given a distributed uncertain database as shown in Table 2, $r = 0.5$, $t = 0.8$, $\epsilon = \theta = 0.1$, $\delta = 0.01$, we first obtain the two parameters of the *AMS-Sketch*, i.e. $m_1 = 5$, $m_2 = 2000$. For the site S_1, when the tuple $\{A(0.8), B(0.2)\}$ arrives, we need to perform 5×2000 samplings. For example, in the first sample, we obtain the item A without update. Thus, in the $AMS-Sketch$ of A, $C_{1,1}^{A,S_1} = 1$. If we obtain A with update in the second sample, we first set $C_{1,1}^{A,S_1} = 2$ and send it to the coordinator. Afterwards, we reset $C_{1,1}^{A,S_1}$ to 0. The remaining sampling processes is similar.

In the coordinator side, we calculate the averages of samples in each row as follows: $\left\{Y_1^A = 0.5, Y_2^A = 0.3, Y_3^A = 0.1, Y_4^A = 0.4, Y_5^A = 0.1\right\}$. We then can obtain that the median is $Y^A = 0.3 < 0.8$, and thus A is not an approximate probabilistic frequent item.

4.3 Correctness analysis

In this subsection, we prove the correctness of the proposed sampling algorithm.

Theorem 2 *(Correctness of the Sketch-based Sampling Algorithm) For an arbitrary item X in the result of Algorithm 5, it satisfies the definition of the approximate probabilistic frequent item with probability at least* $1 - \delta$.

Proof According to the aforementioned algorithm, we know that $C_{i,j}^{X,l}$ is one of the $m_1 \times m_2$ sampled counters of an item X at a local site S_l. The latest uploaded value of $C_{i,j}^{X,l}$ is denoted as $\bar{C}_{i,j}^{X,l}$. Since a local site S_l always uploads an sampled count to the coordinator with probability p, the coordinator actually receives $\bar{C}_{i,j}^{X,l} \leq C_{i,j}^{X,l}$. Let $\Delta_{i,j}^{X,l} = C_{i,j}^{X,l} - \bar{C}_{i,j}^{X,l}$, which is a random variable of the underestimated value between $C_{i,j}^{X,l}$ and $\bar{C}_{i,j}^{X,l}$ and follows the geometric distribution with probability p. Hence, in the coordinator, the global underestimated error of the {i,j}-th counter of X is $\Delta_{i,j}^{X} = \sum_{l=1}^{k} \Delta_{i,j}^{X,l}$, where k is the number of local sites. Thus, we know that each $\zeta_{i,j}^{X}$ follows Bernoulli distribution according to the corresponding global underestimated error $\Delta_{i,j}^{X}$. Hence, we can obtain the expectation and variance of each Y_i^X,

$$\begin{aligned} E\left[Y_i^X\right] &= \frac{\sum_{j=1}^{m_2} E\left[\zeta_{i,j}^X\right]}{m_2} = \mu, \\ Var\left[Y_i^X\right] &= \frac{\sum_{j=1}^{m_2}\left(Var\left[\zeta_{i,j}^X\right]\right)}{m_2^2} \leq \frac{1}{m_2} \end{aligned} \tag{11}$$

Hence, we can obtain the following bound,

$$Pr\left(|Y_i^X - \hat{Y}_i^X| \leq \epsilon\right) \geq 1 - \frac{1}{4\epsilon^2 m_2} \ (\textit{Chebyshev Inequality}) \tag{12}$$

Let I be the number that $|Y_i^X - \hat{Y}_i^X| \geq \epsilon$ holds, which means that X actually violates the definition of $APFI$. Since Y^X is the median of m_1 Y_i^X $(1 \leq i \leq m_1)$, we focus on the whether probability of $I > \frac{m_1}{2}$ is fewer than δ. If so, the theorem holds. Namely,

$$Pr\left(I > \frac{m_1}{2}\right) = Pr\left(I - \frac{l}{4} \geq \frac{l}{4}\right) \leq e^{\frac{9l}{8}} < \delta \ (\textit{Chernoff Inequality}) \tag{13}$$

Hence, the definition of $APFI$ is obeyed with probability at least 1-δ. The theorem holds. $\square$

5 Experimental study

In this section, we report the experimental results on the performance of our proposed algorithms, in terms of communication cost and efficiency. In order to conduct a fair comparison, we build up a test bed on a cluster of 16 servers. Every server has 4 Intel(R) Xeon(R) E5–2650 CPUs of 2.0 GHz, each of which has 4 cores and supports 16 threads, 12 GB memory and 1 TB hard disk storage. The running operating system is 2.6.35-22–server #35-Ubuntu SMP. We employed the MPI implementation for consideration of time efficiency. MPICH2–1.4.1p1 is adopted and the compiler switch O3 is on. Moreover, all the algorithms were implemented in C++.

5.1 Experimental setting

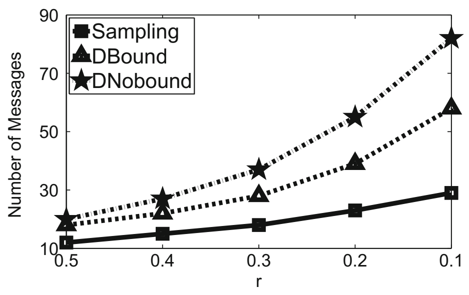
We evaluate our algorithms on both a real data set and a synthetic data set. The real data set is from the SAMOS project [35], which has been used in previous studies of distributed probabilistic data [37]. The raw data of this dataset comes from the research vessel *Wecoma* and includes 11.79 million records, which are generated by the oceanographic measurements from March to November in 2010. Each record contains time and date, wind direction, wind speed, sound speed, and temperature measurements.

For the synthetic data set, we choose a classical deterministic benchmark, *gazelle*, and assign the probability generated from Gaussian distribution to each item. Assigning probability to deterministic database to generate synthetic uncertain test data is widely accepted in the current community [1, 27, 38]. The *gazelle* dataset comes from the click-stream data of a small dot-com company called Gazelle.com. This dataset was used in the KDD-Cup 2000 competition (available on www.ecn.purdue.edu/KDDCUP). For each record in this dataset, each item is randomly assigned a probability which follows the Gaussian distribution (mean = 0.9, variance = 0.1) and the probabilities of the items in each row are normalized to satisfy the requirement of x-relation model. Furthermore, each record has a local-site-S_i choice, where i is randomly selected in the range of 1 to 10. Since the flat model is used, server-to-client communication is broadcast, while client-to-server communication is unicast. The server-to-client broadcast counts as one single message, regardless of the number of clients. The default values of the key parameters are: $r = 0.3$, $\epsilon = 0.1$, $t = 0.8$, $\theta = 0.1$ and $k = 10$.

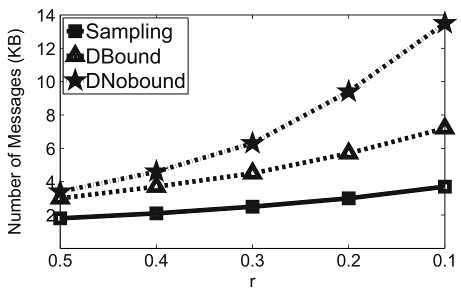
5.2 Communication cost

In this subsection, we compare the communication costs of three algorithms, namely *DNoBound*, *DBound* and *Sampling*. *DNoBound* represents Algorithm 4 without any bounding techniques; *DBound* and *Sampling* represent the complete Algorithm 4 and Algorithm 5, respectively.

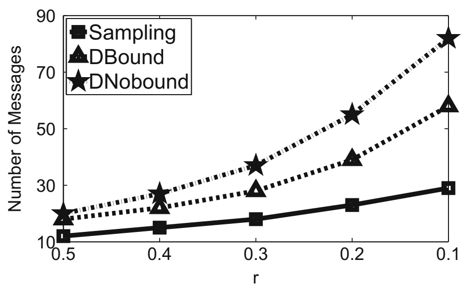
Effect of r First, we demonstrate the communication costs of *DNoBound*, *DBound* and *Sampling* when we vary r from 0.5 to 0.1, as shown in Figures 3a to d. Both the number



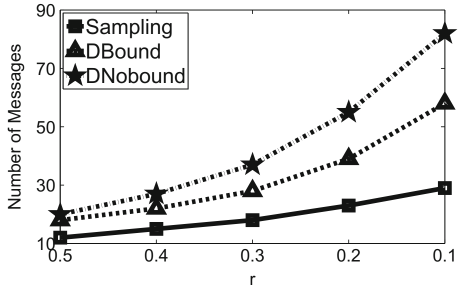
(a) SAMOS: Message vs. R



(b) SAMOS: Byte vs. R



(c) Gazelle: Message vs. R



(d) Gazelle: Byte vs. R

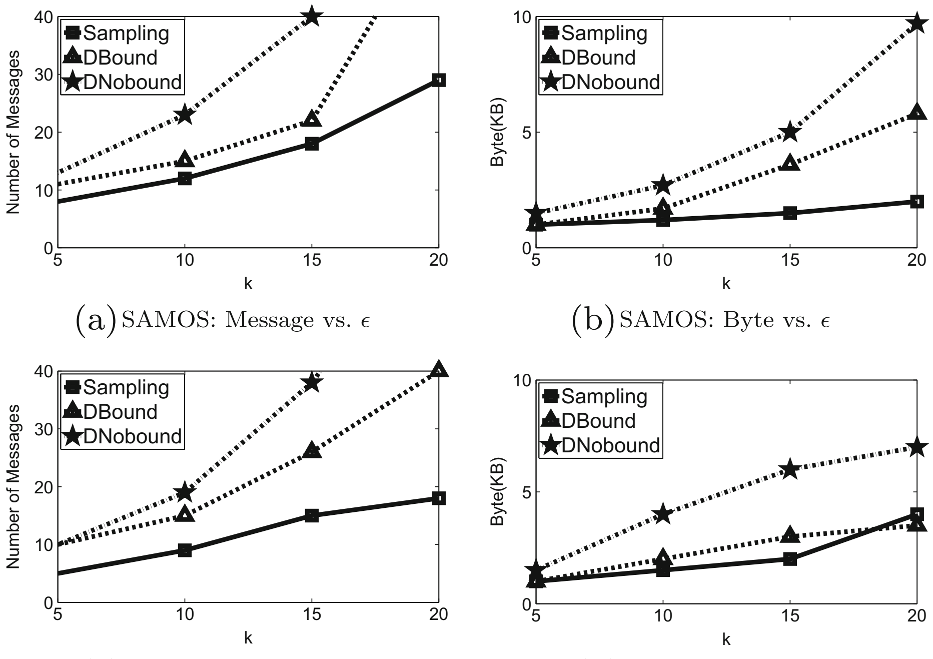
Figure 3 Performance of communication cost with varying r

of messages and bytes increase for all algorithms when r increases, because a smaller minimum support ratio would lead to a larger number of items qualified as frequent, and hence more communication cost. The three competing algorithms are run on both SAMOS and Gazelle, and we show the communication cost in both terms of the number of messages and bytes. Clearly, in all cases, *Sampling* outperforms the others. In addition, *DBound* significantly outperforms *DNoBound*.

Effect of ϵ When ϵ changes, Figures 4a to d show the communication costs of various methods.*DNoBound* still performs the worst among the three. An important finding is that, with the increase of ϵ, *Sampling* becomes less advantageous compared to *DBound*. When ϵ is large enough, *DBound* becomes even more efficient than *Sampling*. The reason is that, the increase of ϵ enhances the pruning powers of the proposed bounding techniques, and hereby many infrequent items are filtered out without inducing any communication cost.

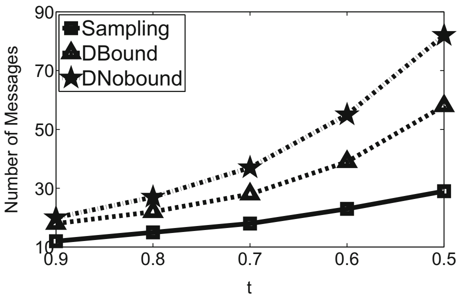
Effect of t We next investigate the impact of the probabilistic threshold. Figures 5a to d show the results of communication cost. When t is changed from 0.9 to 0.5, *DNoBound* benefits the most as its communication cost in terms of both the number of messages and bytes is decreasing. *DBound* still consumes much fewer messages and bytes than *DNoBound* does, and *Sampling* performs the best in all cases.

Effect of θ As shown in Figures 6a to d, the ranking of the three algorithms remains the same - *Sampling* is still the best and *DBound* remains the worst. In addition, we find that the change of θ has little influence on the performances of the algorithms.

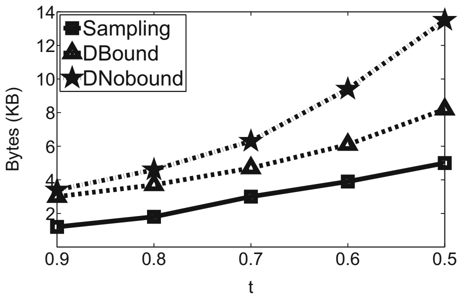


(a) SAMOS: Message vs. ϵ (b) SAMOS: Byte vs. ϵ (c) Gazelle: Message vs. ϵ (d) Gazelle: Byte vs. ϵ

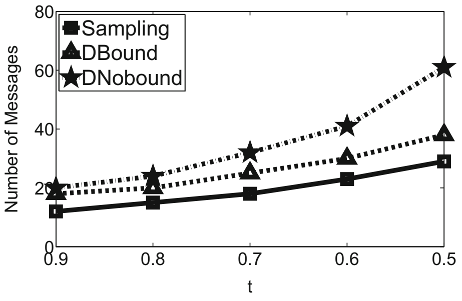
Figure 4 Performance of communication cost with varying ϵ



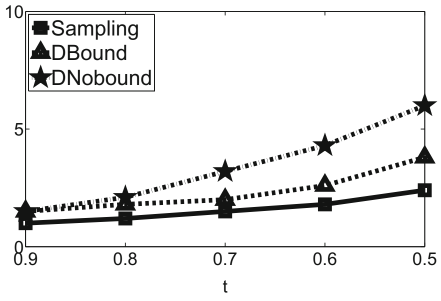
(a) SAMOS: Message vs. t



(b) SAMOS: Byte vs. t



(c) Gazelle: Message vs. t



(d) Gazelle: Byte vs. t

Figure 5 Performance of communication cost with varying t

Effect of k Finally, we investigate the impact of the number of sites. Figures 7a and d show the results. We can observe a linear correlation between the number of messages and k, where *Sampling* consistently performs the best. In addition, all algorithms send more bytes as k increases, and both *Sampling* and *DBound* send much fewer bytes than *DNoBound* does.

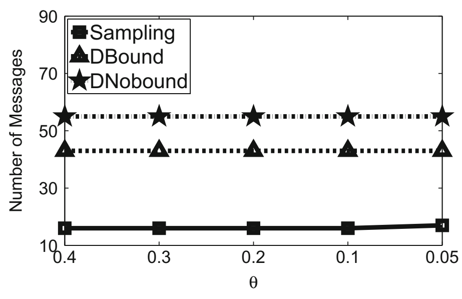
Conclusion The sampling algorithm (Algorithm 5) beats the deterministic algorithms (Algorithm 4 and Algorithm 4 without bounding techniques) in terms of both the number of communicated messages and the communicated Bytes. In other words, the sampling algorithm has smaller communication cost than that of deterministic algorithms.

6 Related work

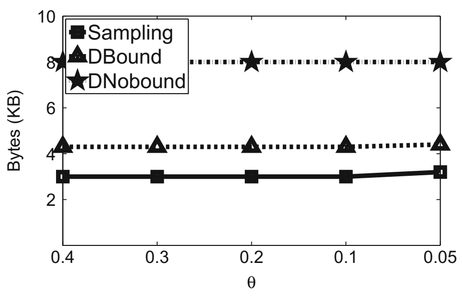
In this section, we review the related work of tracking over distributed data in both deterministic and probabilistic environments.

6.1 Tracking in distributed probabilistic data

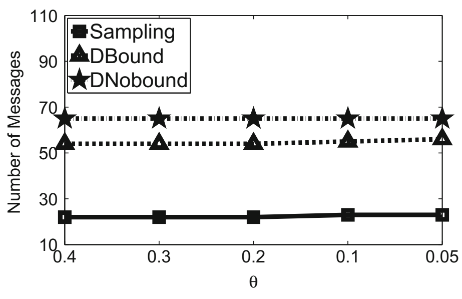
The most closely related researches to our work are about querying over distributed probabilistic data. Li et al. [27] propose the first work of ranking queries in distributed probabilistic data. Recently, a communication-efficient solution for probabilistic threshold-based distributed skyline query has been proposed in [18].



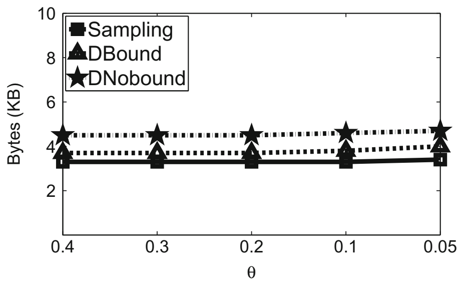
(a) SAMOS: Message vs. θ



(b) SAMOS: Byte vs. θ



(c) Gazelle: Message vs. θ

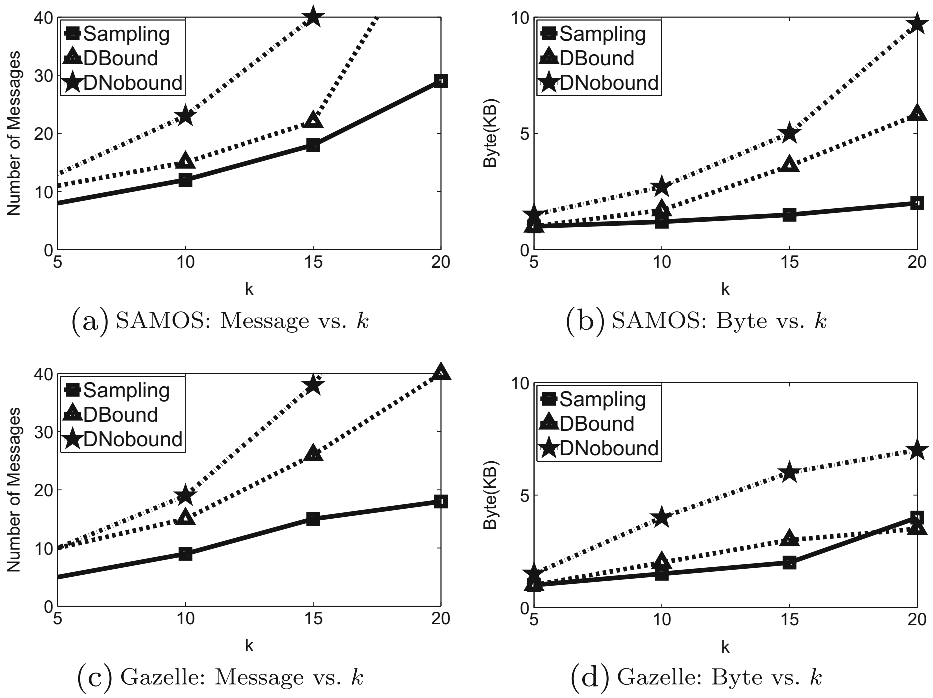


(d) Gazelle: Byte vs. θ

Figure 6 Performance of communication cost with varying θ

In particular, a close work, querying probabilistic top-k frequent items in distributed probabilistic data [44], has been proposed. This work aims to find k probabilistic items with the largest frequent probabilities from distributed probabilistic data. Similar to [27], this work also focuses on maintaining a ranking list of top-k frequent items with minimal communication cost. There are three main differences between [44] and our work. First, the problem definitions are different. [44] only detects top-k probabilistic frequent items; however, our work is to find all probabilistic frequent items. Thus, it is infeasible to directly extend the methods of [44] to track all probabilistic frequent items. Our experimental studies also prove that solution in [44] is not suitable for our problem. Second, [44] only provides a naive pruning method without effective pruning and bounding techniques. Third, there is no comprehensive theoretical analysis on communication cost in [44]. Tang et al. [37] have also studied the problem of threshold-based monitoring in distributed probabilistic data, which is to aggregate constraint monitoring on distributed uncertain data, and provide an efficient and effective solution to reduce the communication and the computational costs.

Besides querying in distributed probabilistic data, another related line of works is finding and mining frequent items(or itemsets) in stand-alone uncertain data. As introduced in Section 1, the definition of a frequent item (or called heavy hitter) over uncertain data has two types of semantic explanations: *expected heavy hitters* [14] and *probabilistic heavy hitter* [47]. In a stand-alone environment, the solutions aim to reduce memory cost rather than communication cost. Moreover, a more complicated and related topic is to discover frequent itemsets over uncertain data. Similar to querying heavy hitters over uncertain stream, frequent itemset over uncertain data has two types of definitions, *expected support-based frequent itemset* [1, 6, 10, 11] and *probabilistic frequent itemset* [5, 36]. Since the number of frequent itemsets may be exponential, most works about mining frequent itemset in



(a) SAMOS: Message vs. k (b) SAMOS: Byte vs. k

(c) Gazelle: Message vs. k (d) Gazelle: Byte vs. k

Figure 7 Performance of communication cost with varying k

uncertain data only handle static uncertain data [38, 40]. They mainly focus on effective index structures or probability distribution-based approximation methods to speed up the mining process. For example, under the definition of expected support-based frequent itemset, UFP-tree [26] and UH-Mine [1] are recently proposed to enhance the performance of the algorithms. Moreover, under the definition of *probabilistic frequent itemset*, the *support* of any itemset actually follows Possion Binomial distribution. According to the probability distribution approximation theory, both Possion distribution-based approximation method [42, 43] and the Normal distribution-based approximation method [7] have been developed. Furthermore, other extended variants [8, 29, 39, 41] of *probabilistic frequent itemset* have been studied recently. Although the previous solutions solve some problems of querying over distributed and stand-alone uncertain data, none of them studies how to track all probabilistic frequent items in distributed probabilistic data.

6.2 Tracking in distributed deterministic data

Another research area, related to our work, is tracking in distributed deterministic data. There are many prior studies. In distributed tracking in deterministic data, the tracked object is a simple count rather than the complicated probability distribution in uncertain scenarios. For the problem of finding frequent items in distributed deterministic data, Yi. et al propose efficient deterministic [45, 46] and randomized algorithms [23]. Moreover, [20] proposes the problem of distributed tracking quantiles in deterministic data, then [13] and [21] provide the efficient deterministic and randomized algorithms, respectively. The efficient solution to track distributed aggregates over sliding-window streams is proposed in

[17]. Recently, [12] provides a comprehensive survey for continuous distributed monitoring models. Although there are a lot of works on tracking in distributed deterministic data, all these solutions are designed for exact data and cannot be extended to probabilistic data directly.

7 Conclusions

In this paper, we focus on the problem of tracking distributed probabilistic frequent items (*TDPF*). In order to reduce the computational cost as much as possible, we propose a comprehensive solution. In our solution, we first design a local threshold-based deterministic algorithm. Especially, two effective pruning rules and a local-threshold-based update mechanism are proposed. Since the deterministic algorithm may encounter high communication and computation cost in the verification phase, we further develop an efficient sketch-based sampling method, which not only reduces the communication cost but also enhances the efficiency through avoiding the high computation burden. Furthermore, the communication costs of all algorithms are theoretically analysed. Finally, the extensive experiments demonstrate the effectiveness and efficiency of the proposed algorithms.

Acknowledgments We are grateful to anonymous reviewers for their constructive comments on this work. This work is supported in part by the Hong Kong RGC Project N_HKUST637/13, National Grand Fundamental Research 973 Program of China under Grant 2014CB340303, NSFC Grant No. 61232018, Microsoft Research Asia Gift Grant, Google Faculty Award 2013, and Microsoft Research Asia Fellowship 2012.

References

1. Aggarwal, C., Li, Y., Wang, J., Wang, J.: Frequent pattern mining with uncertain data. In: Proc. of SIGKDD, pp. 29–38 (2009)
2. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases In: Proc. of SIGMOD, pp. 207–216 (1993)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proc. of VLDB, pp. 487–499 (1994)
4. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. J. Comput. Syst. Sci. **58**(1), 137–147 (1999)
5. Kriegel, T., Bernecker, H., Renz, M., Verhein, F., Zufle, A.: Probabilistic frequent itemset mining in uncertain databases. In: Proc. of SIGKDD, pp. 119–128 (2009)
6. Calders, T., Garboni, C., Goethals, B.: Efficient pattern mining of uncertain data with sampling. In: Proc. of PAKDD, pp. 480–487 (2010)
7. Calders, T., Garboni, C., Goethals, B.: Approximation of frequentness probability of itemsets in uncertain data. In: Proc. of ICDM, pp. 749–754 (2010)
8. Chen, L., Liu, C., Zhang, C.: Mining probabilistic representative frequent patterns from uncertain data. In: Proc. of SDM, pp. 73–81 (2013)
9. Chernoff, H.: A measure of the asymptotic efficiency for tests of a hypothesis based on the sum of observations. Ann. Math. Stat. **23**, 493–509 (1952)
10. Chui, C., Kao, B.: A decremental approach for mining frequent itemsets from uncertain data. In: Proc. of PAKDD, pp. 64–75 (2008)
11. Chui, C., Kao, B., Hung, E.: Mining frequent itemsets from uncertain data. In: Proc. of PAKDD, pp. 47–58 (2007)
12. Cormode, G.: The continuous distributed monitoring model. SIGMOD Record **42**(1), 5–14 (2013)
13. Cormode, G., Garofalakis, M., Muthukrishnan, S., Rastogi, R.: Holistic aggregates in a networked world: distributed tracking of approximate quantiles. In: Proc. of SIGMOD, pp. 25–36 (2005)

14. Cormode, G., Garofalakis, M.: Sketching probabilistic data streams. In: Proc. of SIGMOD, pp. 281–292 (2007)
15. Cormode, G., Hadjieleftheriou, M.: Finding frequent items in data streams. Proc. of VLDB Endow. **1**(2), 1530–1541 (2008)
16. Cormode, G., Muthukrishnan, S., Yi, K., Zhang, Q.: Continuous sampling from distributed streamss. J. ACM **59**(2), 10 (2012)
17. Cormode, G., Yi, K.: Tracking distributed aggregates over time-based sliding windows. In: Proc. of SSDBM, pp. 416–430 (2012)
18. Ding, X., Jin, H.: Efficient and progressive algorithms for distributed skyline queries over uncertain data. IEEE Trans. Knowl. Data Eng. **24**(8), 1448–1462 (2012)
19. Dong, X., Halevy, A., Yu, C.: Data integration with uncertainty. In: Proc. of VLDB, pp. 687–698 (2007)
20. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. In: Proc. of SIGMOD, pp. 58–66 (2001)
21. Huang, Z., Wang, L., Yi, K., Liu, Y.: Sampling based algorithms for quantile computation in sensor networks. In: Proc. of SIGMOD, pp. 745–756 (2011)
22. Huang, Z., Yi, K., Liu, Y., Chen, G.: Optimal sampling algorithms for frequency estimation in distributed data. In: Proc. of INFOCOM, pp. 1997–2005 (2011)
23. Huang, Z., Yi, K., Zhang, Q.: Randomized algorithms for tracking distributed count, frequencies, and ranks. In: Proc. of PODS, pp. 295–306 (2012)
24. Karp, R., Shenker, S., Papadimitriou, C.: A simple algorithm for finding frequent elements in streams and bags. ACM Trans. Database Syst. **28**(1), 51–55 (2003)
25. Keralapura, R., Cormode, G., Ramamirtham, J.: Communication-efficient distributed monitoring of thresholded counts. In: Proc. of SIGMOD, pp. 289–300 (2006)
26. Leung, C., Mateo, M., Brajczuk, D.: A tree-based approach for frequent pattern mining from uncertain data. In: Proc. of PAKDD, pp. 653–661 (2008)
27. Li, F., Yi, K., Jestes, J.: Ranking distributed probabilistic data. In: Proc. of SIGMOD, pp. 361–374 (2009)
28. Li, M., Liu, Y.: Underground coal mine monitoring with wireless sensor networks. ACM Trans. Sensor Netw. **5**(2), 10 (2009)
29. Liu, C., Chen, L., Zhang, C.: Summarizing probabilistic frequent patterns: a fast approach. In: Proc. of SIGKDD, pp. 527–535 (2013)
30. Liu, Y., Liu, K., Li, M.: Passive diagnosis for wireless sensor networks. IEEE/ACM Trans. Netw. **18**(4), 1132–1144 (2010)
31. Manku, G., Motwani, R.: Approximate frequency counts over data streams. In: Proc. of VLDB, pp. 346–357 (2002)
32. Metwally, A., Agrawal, D., Abbadi, A.: A simple algorithm for finding frequent elements in streams and bags. ACM Trans. Database Syst. **31**(3), 1095–1133 (2006)
33. Misra, J., Gries, D.: Finding repeated elements. Sci. Comput. Program. **2**(2), 143–152 (1982)
34. Schoute, F.: Dynamic frame length ALOHA. IEEE Trans. Commun. **31**(4), 565–568 (1983)
35. SAMOS: Shipboard Automated Meteorological and Oceanographic System. http://samos.coaps.fsu.edu/
36. Sun, L., Cheng, R., Cheung, D., Cheng, J.: Mining uncertain data with probabilistic guarantees. In: Proc. of SIGKDD, pp. 273–282 (2010)
37. Tang, M., Li, F., Phillips, J., Jestes, J.: Efficient threshold monitoring for distributed probabilistic data. In: Proc. of ICDE, pp. 1120–1131 (2012)
38. Tong, Y., Chen, L., Cheng, Y., Yu, P.: Mining frequent itemsets over uncertain databases. Proc. VLDB Endow. **5**(11), 1650–1661 (2012)
39. Tong, Y., Chen, L., Ding, B.: Discovering threshold-based frequent closed itemsets over probabilistic data. In: Proc. of ICDE, pp. 270–281 (2012)
40. Tong, Y., Chen, L., Yu, P.: UFIMT: an uncertain frequent itemset mining toolbox. In: Proc. of SIGKDD, pp. 1508–1511 (2012)
41. Tong, Y., Zhang, X., Cao, C., Chen, L.: Efficient probabilistic supergraph search over large uncertain graphs. In: Proc. of CIKM, pp. 809–818 (2014)
42. Wang, L., Cheng, R., Cheung, D., Lee, S., Cheng, R.: Accelerating probabilistic frequent itemset mining: a model-based approach. In: Proc. of CIKM, pp. 429–438 (2010)
43. Wang, L., Cheung, D., Cheng, R., Lee, S., Yang, X.: Efficient mining of frequent item sets on large uncertain databases. IEEE Trans. Knowl. Data Eng. **24**(12), 2170–2183 (2012)
44. Wang, S., Wang, G., Chen, J.: Distributed frequent items detection on uncertain data. In: Proc. of ADMA, pp. 509–520 (2010)
45. Yi, K., Zhang, Q.: Optimal tracking of distributed heavy hitters and quantiles. In: Proc. of PODS, pp. 167–174 (2009)

46. Yi, K., Zhang, Q.: Optimal tracking of distributed heavy hitters and quantiles. Algorithmica **65**(1), 206–223 (2013)
47. Zhang, Q., Li, F., Yi, K.: Finding frequent items in probabilistic data. In: Proc. of SIGMOD, pp. 819–832 (2008)
48. Zikopoulos, P., Eaton, C., Zikopoulos, P.: Understanding Big Data Analytics for Enterprise Class Hadoop and Streaming Data. McGraw-Hill Press (2011)