Aggregate Computation over Data Streams

Xuemin Lin and Ying Zhang

The University of News South Wales, NSW 2052, Australia
{lxue, yingz}@cse.unsw.edu.au

Abstract. Nowadays, we have witnessed the widely recognized phenomenon of high speed data streams. Various statistics computation over data streams is often required by many applications, including processing of relational type queries, data mining and high speed network management. In this paper, we provide survey for three important kinds of aggregate computations over data streams: frequency moment, frequency count and order statistic.

1 Introduction

In recent years, we have witnessed the widely recognized phenomenon of high speed data streams. A data stream is a massive real-time continuous sequence of data elements. The typical applications include sensor network, stock tickers, network traffic measurement, click streams and telecom call records. The main challenge of these applications is that the data element arrives continuously and the volume of the data is so large that they can hardly be stored in the main memory (even on the local disk) for online processing, and sometimes the system has to drop some data elements due to the high arriving speed. The data in the traditional database applications are organized on the hard disk by the Database Management System(DBMS) so the queries from the users can be answered by scanning the indices or the whole data set. Considering of the characteristics of the stream applications, it is not feasible to simply load the arriving data elements onto the DBMS and operate on them because the traditional DBMS's are not designed for rapid and continuous loading of individual data element and they do not directly support continuous queries that are typical of data stream applications [6]. Therefore, in order to support the emerging data stream applications, many works on data stream systems and related algorithms have been done by researchers in various communities and it still remains an active research area nowadays.

As mentioned in [6], following characteristics make data streams different from the conventional relational models :

– The data elements in the stream arrive online and the system has no control over the order in which data elements arrive to be processed, either within a data stream or across data streams. Moreover, the system can not predicate the arriving rate of the data elements.
– Data streams are potentionally unbounded in size. The stream elements are usually relational tuples, but sometimes might be semi-structured data like XML and HTML documents or more complex objects.

– Once an element from data streams has been processed, it is discarded or archived. Then it can not be retrieved easily unless it is explicitly stored in the main memory, which typically is small relative to the size of the data stream.

there has been tremendous progress in building Data Stream Manage Systems(DSMSs) as evidenced by many emerging DSMSs like NiagaraCQ [17], STREAM [86, 6], statStream [98], Gougar [94], Aurora [14], Telegraph [59], Borealis [2], Gigascope [32] etc.

Besides the works of building Data Stream Manage Systems (DSMS's), various data stream algorithms have been proposed from various communities like database, network, computation theory and multimedia. Among various computations over data streams, the aggregate computation plays an important role in many realistic applications such as network traffic management system and sensor network. Following are some important aggregate queries studied in the network traffic management system:

– How much HTTP traffic went on a link today from a given range of IP address? This is an example of a slice and dice query on the multidimensional time series of the flow traffic log [81].
– Find out the number of flows (with or without aggregation) which exceed a threshold T [39]. In many applications, the knowledge of these large flows is suffice.
– What are the $1-\phi$,$1-\phi^2$, $1-\phi^3$ $\ldots 1-\phi^k$ $(0<\phi<1)$ quantiles of the TCP round trip times to each destination? This is important information to gauge the performance of the network in details [26].
– Identifying the *superspreaders* in the network. A superspreader is defined to be a host that contacts at least a given number of distinct destinations within a short time period. Superspreaders could be responsible for fast worm propagation, so it is important to detect them at early stage [91].

The rest of the paper is organized as follows. Section 2 presents the computational model of the data stream algorithms. Then three important aggregate computations over data stream are introduced in sections 3,4 and 5. Particularly, section 3 and section 4 give survey on the frequency moment and frequency count computation over data streams respectively. In section 5, we first introduce the rank computation over data stream with uniform and relative error metrics. Then the top-k computation follows. Some future work on the aggregate computation over data streams are proposed in section 6.

2 Computation Model of Data Stream Algorithms

Because of the unique characteristics of data stream applications, following issues are critical for data stream algorithms:

2.1 Processing Time

In many data stream applications, data elements arrive at a high speed so it is essential for the system to reduce the per record processing time. Otherwise the system might get congested and many elements will be dropped without being processed since usually there is no enough space to keep all of the elements. The arriving rate might burst in some applications, so the consideration of buffering and load-shedding is also required. Likewise, the query response time is another critical issue as a short response time is one of key requirements in many real time data stream applications like network monitoring and stock data analysis.

2.2 Space Usage

Since the size of the data stream is potentially unbounded, it is infeasible to keep all of the stream data elements. Moreover, although the processing time of secondary storage device has been significantly improved in recent years, it might be unacceptable even the system simply keeps every incoming stream element. So many steam algorithms are confined to the main memory without accessing the disk. Consequently, only a synopsis of the data stream can be kept to support user queries. Usually the space used is at most poly-logarithmic in the data size. Sampling, histogram, wavelet and sketch are widely used techniques to *summarize* the stream data.

2.3 Accuracy

It has been shown that in order to get exact answers for some important complex statistics like median and the number of distinct value, a linear space is required. As the synopsis maintained by the system must be very small in size, usually poly-logarithmic in the size of the data stream, the approximation is a key ingredient for stream algorithms. In many applications the exact answer is not crucial, so an approximate answer is sufficient. The system needs to make a trade-off between accuracy and storage space. Hopefully, the algorithm's performance in terms of accuracy will decrease gracefully when there is less memory available.

3 Frequency Moment Computation

Suppose a data stream consists of elements $\{a_1, a_2, \ldots, a_m\}$ which arrive sequentially and a_j is a member of $U = \{1, 2, \ldots, n\}$. Let f_i denote the number of occurrences of i in the data stream. The k-th frequency moment of the data set, denoted by F_k, is defined by $\sum_{i=1}^{n} f_i^k$. Frequency moments play an important role in many applications as they can capture the demographic information of the data set. Particularly, F_0 is the number of distinct elements appearing in the data sequence and F_1 is the length of the sequence. While F_2 is the self-join size (also called surprise index) of the data set and F_∞ is the maximal f_i. In [4], Alon *et al.* present the seminal work to study the problem of frequency moment computation against data streams. It is shown that the exact computation of F_k ($k \neq 1$) needs space linear to the data set size. A general frame work is proposed to estimate the F_k in [4] and many works [21,62,47] have been done in the literature to improve the space(time) efficiency and theoretical bounds. The range efficient computation of frequency moment is studied in [11] and their result is improved by Buriol *et al.* in [13]. The problem of frequency moment computation over sliding window is studied in [35].

Compared with other F_ks, much more attention has been given to the computation of F_0 because of its wide applications. Flajolet and Martin [46] develop the well known *FM* algorithm to estimate the F_0 of the dataset with one scan. A new algorithm is proposed by Durand and Flajolet [37] to improve the space complexity. As algorithms in [46, 37] assume the existence of hash functions with some ideal properties which are hard to construct in practise, Alon *et al.* build on similar technique but only require random pairwise independent hash functions. An adaptive distinct sampling technique is developed by Gibbons *et al.* in [49, 48]. In [10], three new algorithms are proposed to improve the space and time efficiency of previous work. In the context of graphic theoretic applications, Cohen [18] develops a size-estimation framework to explore the closure and reachability of a graph. The *linear counting* algorithm is proposed by Whang *et al.* in [92] based on the *bitmap counting* technique to estimate the cardinality in the database applications. The result is further improved in [41] based on an *adaptive bitmap* technique. Moreover, range efficient algorithm for estimating F_0 on stream data is proposed by Yossef *et al.* in [11]. Recently, [1] improves the time efficiency of the algorithm. And the same problem is investigated under sliding window model by [35] and [50].

4 Frequency Counting

Frequency counting is one of the most basic statistics of a data set because it can mark the most influential part of elements, especially in the skewed data distribution. It has many applications including network monitoring, traffic management, click stream monitoring, telephone call recording and sensor readings. Ideally, we would like to keep track of the top k elements with the highest frequency for desired value of k (top-k elements) or the elements with frequency exceeding a pre-given threshold (*heavy hitters*). For simplicity, we call them *frequent* elements. Exact computation of *frequent* elements against data stream in a small space (sub-linear to N) is infeasible. Rather, various approximate algorithms are proposed to address this problem in the context of data streams.

Misra *et al.* [78] present the first deterministic algorithm for finding ϵ-approximate frequent elements, which uses $O(\frac{1}{\epsilon})$ space and $O(1)$ amortised processing time. Recently [36] and [65] improve the algorithm by reducing the processing time to $O(1)$ in the worst case. Their algorithms guarantee to find all of the frequent candidates in the first pass, but the second pass is required to identify real frequent ones from the candidates. In many data stream applications, it is infeasible to rescan the data. By combining the hashing and sampling techniques, Estan and Verghese [40] present a novel algorithm to identify flows which exceed a pre-defined threshold. In [74], a deterministic algorithm called *lossy counting* is presented for ϵ-approximate frequency elements. Only one pass is required in their algorithm and the worst working space used is $O(\frac{1}{\epsilon}\log(\epsilon N))$. An adaptive sampling algorithm called *sticky sampling* is also proposed in [74]. Although *stick sampling* only uses $O(\frac{1}{\epsilon})$ space in the worst case, it is shown in [74] that *lossy counting* is more space efficient in practise. Metwally *et al.* [77] present *space saving* algorithm which can support approximation of top-k elements and *heavy hitters* with a unified data structure called *Stream Summary.*

Recently, [53] investigates how to efficiently compute the frequent elements in the data stream with the graphics processors. And Bandi *et al.* [9] study the problem under a special networking architectures called Network Processing Units(NPUs). Base on the *lossy counting* [74] and *space saving* [77] techniques, two TCAM-conscious algorithms are proposed to provide efficient solutions.

The algorithms above can not work under *Turnstile Model.* Consequently some *sketch* based algorithms are proposed to address this problem. As a variance of *AMS* sketch in [4], the *count sketch* technique is introduced by Charikar *et al.* [16] to find k elements whose frequencies are at least $(1-\epsilon)$ times the frequency of the k-th most frequent elements, with probability at $1-\delta$ and space $O(\frac{1}{\epsilon^2}\log\frac{n}{\delta})$. In [28], a novel algorithm called *group test* is proposed by Cormode *et al.* to further reduce the space requirement by re-examining the previous algorithm in [16]. Based on the main idea of the *bloom filter*, which is a classical data structure to support membership queries with certain probabilistic guarantees, [39, 19] extend the *bloom filter* to find frequent elements in the data stream. The *Minimal Increase* and *Recurring Miminum* techniques are introduced to improve the accuracy of their algorithm. Recently, following the basic idea of previous algorithms, a new *count min* sketch is presented by Cormode *et al.* in [29]. Their algorithm significantly improve theoretical space efficiency of the previous results by reducing a factor of $O(\frac{1}{\epsilon})$. Their essential idea is to estimate the frequency of each element in the domain by maintaining hash based multiple counters. A similar sketch called *hCount* is independently developed by Jin *et al.* in [63].

The problem has been studied in various applications and many new algorithms are introduced to support different scenarios.

- As the hierarchy structure is widely employed in various online applications such as network management, text mining and XML data summarisation, Cormode *et al.* [24] develop novel algorithm to find out the *hierarchical heavy hitters(HHHs)* based on a layered structure. In [25], the algorithm is extended by the same authors to support identifying the *HHHs* for the hierarchical multidimensional data. The lower bound of the space complexity for this problem is studied in [60].
- [52] proposes an efficient algorithm to find the frequent elements against the sliding windows over online packet streams. Based on a layered structure, new algorithm with bounded space is introduced by Arasu and Manku in [5]. Recently, improvement work is done by Lee and Ting in [68]. Both [5] and [68] study the problem over the variable sliding window as well.
- With the development of the sensor network, various efficient algorithms have been proposed to collect or monitor the frequent elements with tree model [85, 73, 72] as well as multipath model [20, 72, 58]. One of the major concern of these work is the communication cost(total cost and maximal cost between any two nodes) for the computation. In addition to finding frequent elements in a *snapshot* fashion, the problem of continuous monitoring

frequent elements in the network also attracts much attention in the literature [7,23,66,31].
- In some applications, it is desirable to find the *distinct frequent elements*. For instance, in order to identify the potential attacks in the network system such as the Distributed Denial of Service(DDoS) and warm, one of the important approaches is to find out the sources that contact many *distinct* destinations, which is called *superspread*. This problem has been studied in a number of papers [41,91,30,8].

5 Order Statistic Computation

Among various statistics, order statistics computation is one of the most challenging, and is employed in many real applications, such as web ranking aggregation and log mining [3,38], sensor data analysis [55], trends and fleeting opportunities detection in stock markets [6,71], and load balanced data partitioning for distributed computation [76,84].

In this section, we will introduce existing works on two kinds of order statistic oriented queries: rank queries and top-k ranked queries. Although the top-k ranked query can be regarded as a special case of rank query where the rank is limited between 1 and k, they have different focuses. For the rank query, we can only provide approximate solution in the context of data stream while the exact solution is required for the top-k ranked query. Moreover, the rank function is pre-given for the rank query problem while usually we need to support *ad-hoc* rank function for the later problem.

5.1 Rank Query

A rank query is essentially to find a data element with a given rank against a monotonic order specified on data elements. And it has several equivalent variations [57,30]. Rank queries over data streams have been investigated in the form of *quantile* computation. A *ϕ-quantile* ($\phi \in (0,1]$) of an ordered set of N data elements is the element with rank $\lceil \phi N \rceil$.

Rank and quantile queries have many applications including query optimization, finding association rule, monitoring high speed networks, trends and fleeting opportunities detection in the stock markets, sensor data analysis, webranking aggregation, log mining and query visualisation etc. Simple statistics such as the mean and variance are both insufficiently descriptive and highly sensitive to data anomalies in real world data distributions, while quantiles can summarize the distribution of massive data more robustly. Several applications employ quantile algorithms as a foundation, like counting inversions [57] and maintaining reverse nearest neighbour aggregates [67] in the context of data streams.

It has been shown in [80] that the space required for any algorithm to compute the exact rank query with p passes is $\Omega(N^{\frac{1}{p}})$, where N is number of elements. Clearly, it is infeasible to do the exact rank computation in data stream applications where data is massive in size and fast in arriving speed. Consequently, approximate computation of rank queries over data stream has receive a great attentions in the recent years [80].

In this subsection, we will introduce the space and time efficient techniques of continuously maintaining data summaries to support the rank(quantile) queries in various data stream models.

Suppose an element x may be augmented to (x, v) where $v = f(x)$ (called "value") is to rank elements according to a monotonic order v and f is a pre-defined function. Without loss of generality, we assume $v > 0$ and the monotonic order is always an increasing order. We study the following rank queries over a data stream S.

Rank Query: given a rank r, find the rank r element in S.

Suppose that r is the given rank in a rank query and r' is the rank of an approximate solution. We could use the constant-based absolute error metric, say $|r - r'| \leq \epsilon$ for any given ϵ. It is immediate that such an absolute error precision guarantee will lead to the space requirement $\Omega(N)$ even for the an offline computation where $N = |S|$. So two kinds of error metrics have been used in the recent works.

Uniform Error. $\frac{r'-r}{N} \leq \epsilon$.
Relative Error. $\frac{r'-r}{r} \leq \epsilon$.

An answer to a rank query regarding r is *uniform ϵ-approximate* if its rank r' has the precision $|r - r'| \leq \epsilon N$. And it is *relative ϵ-approximate* if its rank r' has the precision $|r - r'| \leq \epsilon r$. In the following part, we will introduce the techniques of continuously maintaining a synopsis over data stream S such that at any time, the synopsis can provide a (relative or uniform) *ϵ-approximate* answer for a given rank query. The focus of the techniques is to minimize the maximal memory space required in such a continuous synopsis maintenance procedure. The processing time per element and query response time are also important issues.

Uniform Error Techniques. In [80], Munro and Paterson present a one pass algorithm to provide the uniform ϵN approximate answer for quantile query. A binary tree structure is employed in their paper and the work space required is $O(\frac{1}{\epsilon}\log^2(\epsilon N))$. Manku *et al.* [75] improve the previous algorithm in terms of the working space. They reduce the constant factor significantly by applying a more sophisticated merge strategy. Then they propose a space efficient randomized algorithm in [76] to further reduce the space bound to $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon\delta})$ by applying an adaptive sampling approach. Then with probability at least $1 - \delta$, their algorithm can achieve ϵN approximation. Moreover, their algorithm can compute the quantiles without the advanced knowledge of the length of the data stream. They also show that further space deduction can be achieved by feeding the sample set to any deterministic quantile algorithm.

Greenwald and Khanna [54] propose the best known deterministic quantile algorithm, called GK algorithm, for the *Cash Register Model*, with $O(\frac{1}{\epsilon}\log(\epsilon N))$ working space in worst case. Following the space reduction framework of [76], a randomized algorithm with space $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon\delta}))$ can be immediately developed. The GK algorithm has been widely employed as a building block in many quantile related works [67, 69, 5, 55].

As to the *Turnstile Model*, Gilbert *et al.* [51] propose the first algorithm to ϵ-approximate the rank query with probability at least $1 - \delta$. Their algorithm is based on estimating *range-sums* of the data stream over *dyadic* intervals with $O(\frac{1}{\epsilon^2} \log^2 |U| \log \frac{\log |U|}{\delta})$ working space, where U is the size of the element domain. The data structure they used for estimating *range-sums* of the data stream is called *Random subset sums sketch*, which can be directly replaced by the *Count-Min sketch* proposed in [29]. Then an immediate improvement over the space complexity follows with $O(\frac{1}{\epsilon} \log^2 |U| \log \frac{\log |U|}{\delta})$, which is the currently best known space bound in the *Turnstile model.* Applications of their algorithms include the telecommunication transaction monitoring and query optimization in the DBMS.

Lin *et al.* [69] propose the first space and time efficient algorithm to continuously maintain order statistics against the *count-based* sliding window model. Their techniques are based on a combination of GK-algorithm [54] and *exponential histogram* technique in [35]; They considered the rank queries over *fixed sliding windows* as well as *variable sliding windows.* And their space bound is $O(\frac{\log \epsilon^2 N}{\epsilon} + \frac{1}{\epsilon^2})$ and $(\frac{1}{\epsilon^2} \log^2(\epsilon N))$ for *fixed sliding windows* and *variable sliding windows* respectively. Based on a more sophisticated interval-tree like structure, Arasu and Manku [5] improve the space bound in [69].

With the development of the sensor network, various statistic computation algorithms on the sensor network have been developed by various communities. Greenwald and Khanna [55] study the problem of power-conserving computation of order statistics in sensor networks. They show that the tree model of the sensor network model is at least as hard as stream model. Their algorithm enforces that the largest load difference between any two nodes will not exceed $O(\log(\epsilon N))$ in order to achieve ϵ-approximation. The maximal load for each node is bounded by $O(\frac{\log^2 n}{\epsilon})$. Shrivastava *et al.* [85] improve the maximal load to $O(\frac{\log n}{\epsilon})$ based on a novel *Q-digest* data structure.

Instead of answering rank queries against a *snapshot* of the data set like [55, 85], Cormode *et al.* [22] investigate the problem of continuous tracking of complex aggregates (e.g quantile) and data-distribution summaries over collections of distributed streams. In order to achieve highly communication- and space-efficient solutions, they combine a local tracking at remote sites and simple prediction models for local site behaviour.

In [70], novel techniques are proposed to efficiently process a massive set of continuous rank queries where the *Continuous Queries* are issued and run continuously to update the query results along with the updates of the underlying datasets.

In [56], Guha *et al.* investigate the importance of the ordering of a data stream, without any assumptions about the actual distribution of the data. The quantile computation is used as a sample application. They prove some theoretical space bounds for the quantile algorithm over the data streams with *adversary* and *completely random* order. And their space efficient technique enforces a finer rank error guarantee $|r - r'| = O(r^{0.5+\epsilon})$. [53] shows how to efficiently compute the quatiles over the data stream with the graphics processors.

Relative Error Techniques. Using the relative error metric to measure approximation is not only of theoretical interest but also very useful in many applications. For instance, as shown in [26], finer error guarantees at higher ranks are often desired in network management. This is because IP traffic data often exhibits skew towards the tail and it is exactly in the most skewed region where user wants relative rank error guarantees, to get more precise information about changes in values. Relative error is also motivated by the problem of approximately *counting inversions* of a data stream [57].

The problem of finding approximate quantiles with relative error guarantee is first studied by Gupta and Zane [57], who develop a one-scan randomized technique with $O(\frac{1}{\epsilon^3}\log^2 N)$ space requirement for approximately counting inversions, by maintaining an order sketch with the relative rank error guarantee ϵ. However, their technique requires advanced knowledge of (an upper bound on) N to do one-scan sampling. This potentially limits its applications. Cormode *et al.* [26] study the related problem of computing *biased quantiles*, that is, the set of quantiles $\Phi = \{\phi_i = \phi_0^i : 1 \leq i \leq k\}$, for a fixed k and some ϕ_0, which are estimated with precision $\epsilon\phi_i N$. [26] gives an algorithm to approximate such biased quantiles with deterministic error guarantees which performs very well against many real data sets. While the problem of computing biased quantiles focuses on the relative rank error guarantee bounded by a minimum quantile $\phi_0^k N$, the rank query addresses relative error guarantees at *all* ranks, no matter how small ϕ is. As shown in [26], the application of their technique to the arbitrary rank queries leads to a linear space requirement $\Omega(N)$ in the worst case; this can render the deterministic technique impracticable in applications where small space usage is imperative.

In [96], We developed a novel, one-scan randomized algorithm ("MR") which guarantees the precision ϵ of relative rank errors with confidence $1-\delta$ and requires $O(\frac{1}{\epsilon^2}\log\frac{2}{\delta}\log\epsilon^2 N)$ space. We also develop an effective one-scan space compression technique. Combined with the above one-scan randomized technique, it leads to a more space-efficient one-scan randomized algorithm ("MRC") which guarantees the average space requirement $O(\frac{1}{\epsilon}\log(\frac{1}{\epsilon}\log\frac{2}{\delta})\frac{\log^{2+\alpha}\epsilon N}{1-1/2^{\alpha}})$ (for $\alpha > 0$), while the worst case space requirement remains $O(\frac{1}{\epsilon^2}\log\frac{2}{\delta}\ \log\epsilon^2 N)$. Recently, Cormode*et al.* [27] develop a novel deterministic algorithm to approximate the rank queries with relative error. The *Q-digest* structure in [85] is extended in their work, and the space required by their algorithm is $O(\frac{\log|U|}{\epsilon}\log\epsilon N)$. As shown in [27], their algorithm outperforms the randomized algorithms. However, their solution is restricted to a fixed value domain U. The space efficient deterministic algorithm with relative error guarantee remains open for the applications where the domain size of the data elements is unlimited.

Duplicate-insensitive. In many data stream applications, duplicates may often occur due to the projection on a subspace if elements have multiple attributes. For example, in the stock market a deal with respect to a particular stock is recorded by the transaction ID (TID), volume (*vol*), and average price (*av*) per share. To study purchase trends, it is important to estimate the number of different types of deals (i.e. deals with the same vol and the same av are regarded as the same type of deal) with their total prices (i.e. *vol*av*) higher (or lower) than a given value. It is also interesting to know the total price (of a deal) ranked as a median, or 25th percentile, or 10th, or 5th percentile, etc. among all different types of deals. These two types of rank queries are equivalent [27,57]; To accommodate such queries, we need to project each deal transaction (TID, *vol*, *av*) on (*vol*, *av*) and then summarize the distribution of **distinct**(*vol*, *av*)s according to a decreasing (or increasing) order of *vol***av*. In this application, the data elements to be summarized are mapped from (TID, *vol*, *av*) to (*vol*, *av*) by the projection. Consequently, any generated duplicates (*vol*, *av*) must be removed to process such rank queries. Moreover, relative (or biased) rank error metrics need to be used to provide more accurate results towards heads (or tails depending on which monotonic order is adopted). Note that the generality of rank queries (quantiles) remains unchanged in this application since two different types of deals (i.e., (*vol*, *av*)s) may also have the same values of *vol***av*. The unique challenge is to detect and remove the effect of duplicates without keeping every element.

Duplicates may also occur when data elements are observed and recorded multiple times at different data sites. For instance, as shown in [26, 30] the same packet may be seen at many tap points within an IP network depending on how the packet is routed; thus it is important to discount those duplicates while summarising data distributions by rank queries (quantiles). Moreover, to deal with possible communication loss TCP retransmits lost packets and leads to the same packet being seen even at a given monitor more than once. Similarly, in order to achieve high fault-tolerance against communication errors in a sensor network a popular mechanism is to send data items by multi-paths [20, 72, 82] which will create duplicates.

In such distributed applications, continuously maintaining order sketches for processing rank queries may be conducted either centrally at one site or at a set of coordinating sites depending on the computing environment and the availability of software and hardware devices. Nevertheless, in those situations a crucial issue is to efficiently and continuously maintain a small space sketch with a precision guarantee at a single site, by discounting duplicates.

The *FM* technique [46] has been first applied in [12, 20, 82] to develop duplicate-insensitive techniques for approximate computing *sum*, *count* (number of sensor nodes), *average* to achieve high communication fault-tolerance.

In [82], Nath *et al.* present a duplicate-insensitive in-network quantile computation algorithm to cope with multi-path communication protocol. For each element, a random number between $[0, 1]$ is drawn, which determines if the element will remain in the quantile sample; this combines with the element ID to remove duplicates generated by the multipass communication. As the uniform sampling technique does not guarantee to draw the same random number for the duplicated element, the technique in [82] can only handle the duplicates generated in communication rather than duplicates in data streams.

In [72], Manjhi, Nath and Gibbons propose an effective adaption paradigm for in-network aggregates computation over stream data with the aim to minimize communication costs and to achieve high fault-tolerance. A duplicate-insensitive technique for approximately computing quantiles can be immediately obtained by a combination of their tree-based approximation technique and the existing *distinct counting* technique in [10]. It can be immediately applied to a single site, where a data stream has duplicated elements, with the uniform precision guarantee $|r' - r| \leq \epsilon n$ by confidence $1 - \delta$ and space $O(1/\epsilon^3 \log 1/\delta \log m)$, where m is the maximal possible number of distinct elements.

In [30], Cormode and Muthukrishnan present a *Distinct range sums* technique by applying the FM [46] technique on the top of their *count-min* sketch [29]. The technique can be immediately used to approximately process rank query with the uniform precision guarantee $|r' - r| \leq \epsilon n$, confidence $1 - \delta$, and space $O(\frac{1}{\epsilon^3} \log \frac{1}{\delta} \log^2 m)$. Independently, Hadjieleftheriou, Byers and Kollios [58] develop two novel duplicate-insensitive techniques based on [85] and [29] to approximately compute quantiles in a distributed environment. Applying their techniques to a single site immediately leads the uniform precision guarantee $|r' - r| \leq \epsilon n$ by confidence $1 - \delta$ and space $O(\frac{1}{\epsilon^3} \log \frac{1}{\delta} \log m)$.

Very recently, we develop the first space- and time- efficient, duplicate-insensitive algorithms [97] to continuously maintain a sketch of order statistics over data stream to enforce relative ϵ-approximation. They not only improve the existing precision guarantee (from uniform ϵ-approximation to relative ϵ-approximation) but also reduce the space from $O(\frac{1}{\epsilon^3} \log \frac{1}{\delta} \log m)$ to $O(\frac{1}{\epsilon^3} \log \frac{1}{\delta} \log m)$ where m is the element domain size.

5.2 Top-k Ranked Query

Instead of finding records with arbitrary rank, in many applications users are typically interested in the k records with highest ranks, where $k \ll N$ and N is the number of records. Moreover, the ranking(preference) function might be proposed by users at query time. Providing efficient answers to such top-k ranked queries has been a quite active topic and has many important applications involving multi-criteria decision making.

In many applications the volume of the dataset is extremely large while users are usually only interested in a limited number of answers regarding to their preference functions, so it becomes necessary to pre-process the data to speed up the performance. Many related works have been done in the literature, and they can be classified into three categories: distributed index [42,43,44,45], view based index [61,95,34] and minimal space index [15,90,87,93,64,79,89]. However, only a few work [79,88,33] investigate the problem in the context of data streams.

In [79], Mouratidis *el al.* study the problem of continuous monitoring of top-k queries over sliding windows. Based on the concept of K-skyband introduced in [83], it is shown that only the tuples in the K-skyband can be answers for any top-k ranked query with monotonic preference function where $k < K$. In [79], elements are indexed by a regular grid in main memory. Two algorithms, *TMA* and *SMA*, are proposed to continuously monitor the top-k answers for those queries. In [79], a tuple can be regarded as a two dimensional data point. One dimension is the score of the tuple (the rank function is pre-given) and another is its timestamp. The *SMA* algorithm is proposed to continuously maintain the K-skyband against the stream data. The K-skyband of a dataset is the points which can be dominated by at most K-1 other points, Clearly skyline is a special instance of skyband with K=1. The basic idea of *SMA* is to maintain a *dominance count*(DC) for each tuple t where the DC is the number of tuples which dominate t, One tuple can be immediately discarded once its DC exceeds K since it will not be touched by any top-k query with $k \leq K$.

In [88], Tao *et al.* show how to continuously maintain the K-skyband against multidimensional data indexed by the R Tree, and the concept of *dominance count* is also employed in [89] as well. With a branch-and-bound search strategy proposed in [88], [89] can efficiently retrieve answers for the top-k ranked queries.

Recently, based on the novel concept of the geometric arrangement, Das *et al.* [33] further improve the efficiency of the top-k algorithms over data streams. Instead of continuously maintaining the K-skyband of the stream data, new tuple pruning algorithm is proposed in the paper such that the cost of minimal candidate set maintenance is significantly reduced.

6 Future Work

Although there are still many problems remaining open for the data stream algorithms, recently a great attention is given to the aggregate computation over probabilistic data streams. In many important applications such as environmental surveillance, market analysis and quantitative economics research, uncertainty is inherent because of various factors including data randomness and incompleteness, limitations of measuring equipments, delayed data updates, etc. Meanwhile, those data are created rapidly so it is worthwhile to investigate various computations over the probabilistic data streams. The main challenge is to design space and time efficient algorithms to handle the uncertain data which might arrive rapidly.

References

1. Aduri, P., Tirthapura, S.: Range efficient computation of f_0 over massive data streams. In: ICDE, pp. 32–43 (2005)
2. Ahmad, Y., Berg, B., Çetintemel, U., Humphrey, M., Hwang, J.-H., Jhingran, A., Maskey, A., Papaemmanouil, O., Rasin, A., Tatbul, N., Xing, W., Xing, Y., Zdonik, S.B.: Distributed operation in the borealis stream processing engine. In: SIGMOD, pp. 882–884 (2005)
3. Ajtai, M., Jayram, T.S., Kumar, R., Sivakumar, D.: Approximate counting of inversions in a data stream. In: STOC, pp. 370–379 (2002)
4. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. In: STOCK, pp. 20–29 (1996)
5. Arasu, A., Manku, G.S.: Approximate counts and quantiles over sliding windows. In: PODS, pp. 286–296 (2004)
6. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: PODS (2002)
7. Babcock, B., Olston, C.: Distributed top-k monitoring. In: SIGMOD, pp. 28–39 (2003)
8. Bandi, N., Agrawal, D., Abbadi, A.E.: Fast algorithms for heavy distinct hitters using associative memories. In: IEEE International Conference on Distributed Computing Systems(ICDCS), p. 6 (2007)
9. Bandi, N., Metwally, A., Agrawal, D., Abbadi, A.E.: Fast data stream algorithms using associative memories. In: SIGMOD, pp. 247–256 (2007)

10. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D., Trevisan, L.: Counting distinct elements in a data stream. In: Randomization and Approximation Techniques, 6th International Workshop, RANDOM, pp. 1–10 (2002)
11. Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Reductions in streaming algorithms, with an application to counting triangles in graphs. In: SODA, pp. 623–632 (2002)
12. Bawa, M., Molina, H.G., Gionis, A., Motwani, R.: Estimating aggregates on a peer-to-peer network. Technical report, Stanford University (2003)
13. Buriol, L.S., Frahling, G., Leonardi, S., Marchetti-Spaccamela, A., Sohler, C.: Counting triangles in data streams. In: PODS, pp. 253–262 (2006)
14. Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., Zdonik, S.B.: Monitoring streams - a new class of data management applications. In: Bressan, S., Chaudhri, A.B., Li Lee, M., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, pp. 215–226. Springer, Heidelberg (2003)
15. Chang, Y.-C., Bergman, L.D., Castelli, V., Li, C.-S., Lo, M.-L., Smith, J.R.: The onion technique: Indexing for linear optimization queries. In: SIGMOD, pp. 391–402 (2000)
16. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 693–703. Springer, Heidelberg (2002)
17. Chen, J., DeWitt, D.J., Tian, F., Wang, Y.: Niagaracq: A scalable continuous query system for internet databases. In: SIGMOD, pp. 379–390 (2000)
18. Cohen, E.: Size-estimation framework with applications to transitive closure and reachability. J. Comput. Syst. Sci. 55(3), 441–453 (1997)
19. Cohen, S., Matias, Y.: Spectral bloom filters. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 241–252 (2003)
20. Considine, J., Li, F., Kollios, G., Byers, J.W.: Approximate aggregation techniques for sensor databases. In: ICDE, pp. 449–460 (2004)
21. Coppersmith, D., Kumar, R.: An improved data stream algorithm for frequency moments. In: SODA, pp. 151–156 (2004)
22. Cormode, G., Garofalakis, M.N.: Sketching streams through the net: Distributed approximate query tracking. In: VLDB, pp. 13–24 (2005)
23. Cormode, G., Garofalakis, M.N., Muthukrishnan, S., Rastogi, R.: Holistic aggregates in a networked world: Distributed tracking of approximate quantiles. In: SIGMOD, pp. 25–36 (2005)
24. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Finding hierarchical heavy hitters in data streams. In: VLDB, pp. 464–475 (2003)
25. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Diamond in the rough: Finding hierarchical heavy hitters in multi-dimensional data. In: SIGMOD, pp. 155–166 (2004)
26. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Effective computation of biased quantiles over data streams. In: ICDE, pp. 20–31 (2005)
27. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In: PODS, pp. 263–272 (2006)
28. Cormode, G., Muthukrishnan, S.: What's hot and what's not: tracking most frequent items dynamically. In: PODS, pp. 296–306 (2003)
29. Cormode, G., Muthukrishnan, S.: An improved data stream summary: The count-min sketch and its applications. In: Farach-Colton, M. (ed.) LATIN 2004. LNCS, vol. 2976, pp. 29–38. Springer, Heidelberg (2004)
30. Cormode, G., Muthukrishnan, S.: Space efficient mining of multigraph streams. In: PODS, pp. 271–282 (2005)

31. Cormode, G., Muthukrishnan, S., Zhuang, W.: What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In: ICDE, p. 57 (2006)
32. Cranor, C.D., Johnson, T., Spatscheck, O., Shkapenyuk, V.: Gigascope: A stream database for network applications. In: SIGMOD, pp. 647–651 (2003)
33. Das, G., Gunoplulos, D., Koudas, N., Sarkas, N.: Ad-hoc top-k query answering for data streams. In: VLDB (2007)
34. Das, G., Gunopulos, D., Koudas, N., Tsirogiannis, D.: Answering top-k queries using views. In: VLDB, pp. 451–462 (2006)
35. Datar, M., Gionis, A., Indyk, P., Motwani, R.: Maintaining stream statistics over sliding windows (extended abstract). In: SODA, pp. 635–644 (2002)
36. Demaine, E.D., López-Ortiz, A., Munro, J.I.: Frequency estimation of internet packet streams with limited space. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 348–360. Springer, Heidelberg (2002)
37. Durand, M., Flajolet, P.: Loglog counting of large cardinalities (extended abstract). In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 605–617. Springer, Heidelberg (2003)
38. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: WWW, pp. 613–622 (2001)
39. Estan, C., Varghese, G.: New directions in traffic measurement and accounting. In: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications(SIGCOMM) (2002)
40. Estan, C., Varghese, G.: New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. ACM Trans. Comput. Syst. 21(3), 270–313 (2003)
41. Estan, C., Varghese, G., Fisk, M.: Bitmap algorithms for counting active flows on high speed links. In: ACM SIGCOMM Conference on Internet Measurement, pp. 153–166 (2003)
42. Fagin, R.: Combining fuzzy information from multiple systems. In: PODS, pp. 216–226 (1996)
43. Fagin, R.: Fuzzy queries in multimedia database systems. In: PODS, pp. 1–10 (1998)
44. Fagin, R.: Combining fuzzy information from multiple systems. J. Comput. Syst. Sci. 58(1), 83–99 (1999)
45. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: PODS (2001)
46. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. J. Comput. Syst. Sci. 31(2), 182–209 (1985)
47. Ganguly, S., Cormode, G.: On Estimating Frequency Moments of Data Streams. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) RANDOM 2007 and APPROX 2007. LNCS, vol. 4627, pp. 479–493. Springer, Heidelberg (2007)
48. Gibbons, P.B.: Distinct sampling for highly-accurate answers to distinct values queries and event reports. In: VLDB, pp. 541–550 (2001)
49. Gibbons, P.B., Tirthapura, S.: Estimating simple functions on the union of data streams. In: SPAA, pp. 281–291 (2001)
50. Gibbons, P.B., Tirthapura, S.: Distributed streams algorithms for sliding windows. In: SPAA, pp. 63–72 (2002)
51. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: How to summarize the universe: Dynamic maintenance of quantiles. In: VLDB, pp. 454–465 (2002)
52. Golab, L., DeHaan, D., Demaine, E.D., López-Ortiz, A., Munro, J.I.: Identifying frequent items in sliding windows over on-line packet streams. In: ACM SIGCOMM Conference on Internet Measurement, pp. 173–178 (2003)

53. Govindaraju, N.K., Raghuvanshi, N., Manocha, D.: Fast and approximate stream mining of quantiles and frequencies using graphics processors. In: SIGMOD, pp. 611–622 (2005)
54. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. In: SIGMOD, pp. 58–66 (2001)
55. Greenwald, M., Khanna, S.: Power-conserving computation of order-statistics over sensor networks. In: PODS, pp. 275–285 (2004)
56. Guha, S., McGregor, A.: Approximate quantiles and the order of the stream. In: PODS, pp. 273–279 (2006)
57. Gupta, A., Zane, F.: Counting inversions in lists. In: SODA, pp. 253–254 (2003)
58. Hadjieleftheriou, M., Byers, J.W., Kollios, G.: Robust sketching and aggregation of distributed data streams. Technical report. Boston University (2005)
59. Hellerstein, J.M., Franklin, M.J., Chandrasekaran, S., Deshpande, A., Hildrum, K., Madden, S., Raman, V., Shah, M.A.: Adaptive query processing: Technology in evolution. IEEE Data Eng. Bull. 23(2), 7–18 (2000)
60. Hershberger, J., Shrivastava, N., Suri, S., Tóth, C.D.: Space complexity of hierarchical heavy hitters in multi-dimensional data streams. In: PODS, pp. 338–347 (2005)
61. Hristidis, V., Koudas, N., Papakonstantinou, Y.: Prefer: A system for the efficient execution of multi-parametric ranked queries. In: SIGMOD, pp. 259–270 (2001)
62. Indyk, P., Woodruff, D.P.: Optimal approximations of the frequency moments of data streams. In: STOCK, pp. 202–208 (2005)
63. Jin, C., Qian, W., Sha, C., Yu, J.X., Zhou, A.: Dynamically maintaining frequent items over a data stream. In: CIKM, pp. 287–294 (2003)
64. Jin, W., Ester, M., Han, J.: Efficient processing of ranked queries with sweeping selection. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 527–535. Springer, Heidelberg (2005)
65. Karp, R.M., Shenker, S., Papadimitriou, C.H.: A simple algorithm for finding frequent elements in streams and bags. ACM Trans. Database Syst. 28, 51–55 (2003)
66. Keralapura, R., Cormode, G., Ramamirtham, J.: Communication-efficient distributed monitoring of thresholded counts. In: SIGMOD, pp. 289–300 (2006)
67. Korn, F., Muthukrishnan, S., Srivastava, D.: Reverse nearest neighbor aggregates over data streams. In: Bressan, S., Chaudhri, A.B., Li Lee, M., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, pp. 814–825. Springer, Heidelberg (2003)
68. Lee, L.K., Ting, H.F.: A simpler and more efficient deterministic scheme for finding frequent items over sliding windows. In: PODS, pp. 290–297 (2006)
69. Lin, X., Lu, H., Xu, J., Yu, J.X.: Continuously maintaining quantile summaries of the most recent n elements over a data stream. In: ICDE, pp. 362–374 (2004)
70. Lin, X., Xu, J., Zhang, Q., Lu, H., Yu, J.X., Zhou, X., Yuan, Y.: Approximate processing of massive continuous quantile queries over high-speed data streams. IEEE Trans. Knowl. Data Eng. 18(5), 683–698 (2006)
71. Manganelli, S., Engle, R.: Value at risk models in finance. In: European Central Bank Working Paper Series No. 75 (2001)
72. Manjhi, A., Nath, S., Gibbons, P.B.: Tributaries and deltas: Efficient and robust aggregation in sensor network streams. In: SIGMOD, pp. 287–298 (2005)
73. Manjhi, A., Shkapenyuk, V., Dhamdhere, K., Olston, C.: Finding (recently) frequent items in distributed data streams. In: ICDE, pp. 767–778 (2005)
74. Manku, G.S., Motwani, R.: Approximate frequency counts over data streams. In: Bressan, S., Chaudhri, A.B., Li Lee, M., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, pp. 346–357. Springer, Heidelberg (2003)

75. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: Approximate medians and other quantiles in one pass and with limited memory. In: SIGMOD, pp. 426–435 (1998)
76. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: Random sampling techniques for space efficient online computation of order statistics of large datasets. In: SIGMOD, pp. 251–262 (1999)
77. Metwally, A., Agrawal, D., Abbadi, A.E.: Efficient computation of frequent and top-k elements in data streams. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 398–412. Springer, Heidelberg (2004)
78. Misra, J., Gries, D.: Finding repeated elements. Sci. Comput. Program. 2(2), 143–152 (1982)
79. Mouratidis, K., Bakiras, S., Papadias, D.: Continuous monitoring of top-k queries over sliding windows. In: SIGMOD, pp. 635–646 (2006)
80. Munro, J.I., Paterson, M.: Selection and sorting with limited storage. Theor. Comput. Sci. 12, 315–323 (1980)
81. Muthukrishnan, S.: Data streams: algorithms and applications. In: SODA, pp. 413–413 (2003)
82. Nath, S., Gibbons, P.B., Seshan, S., Anderson, Z.R.: Synopsis diffusion for robust aggregation in sensor networks. In: SenSys, pp. 250–262 (2004)
83. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. 30(1), 41–82 (2005)
84. Poosala, V., Ioannidis, Y.E.: Estimation of query-result distribution and its application in parallel-join load balancing. In: VLDB, pp. 448–459 (1996)
85. Shrivastava, N., Buragohain, C., Agrawal, D., Suri, S.: Medians and beyond: new aggregation techniques for sensor networks. In: SenSys, pp. 239–249 (2004)
86. STREAM stream data manager, `http://www-db.stanford.edu/stream/sqr`
87. Tao, Y., Hadjieleftheriou, M.: Processing ranked queries with the minimum space. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 294–312. Springer, Heidelberg (2006)
88. Tao, Y., Hristidis, V., Papadias, D., Papakonstantinou, Y.: Branch-and-bound processing of ranked queries. Inf. Syst. 32(3), 424–445 (2007)
89. Tao, Y., Xiao, X., Pei, J.: Efficient skyline and top-k retrieval in subspaces. IEEE Trans. Knowl. Data Eng (to appear, 2007)
90. Tsaparas, P., Palpanas, T., Kotidis, Y., Koudas, N., Srivastava, D.: Ranked join indices. In: ICDE, pp. 277–288 (2003)
91. Venkataraman, S., Song, D.X., Gibbons, P.B., Blum, A.: New streaming algorithms for fast detection of superspreaders. In: NDSS (2005)
92. Whang, K.-Y., Zanden, B.T.V., Taylor, H.M.: A linear-time probabilistic counting algorithm for database applications. ACM Trans. Database Syst. 15(2), 208–229 (1990)
93. Xin, D., Chen, C., Han, J.: Towards robust indexing for ranked queries. In: VLDB, pp. 235–246 (2006)
94. Yao, Y., Gehrke, J.: The cougar approach to in-network query processing in sensor networks. SIGMOD Record 31(3), 9–18 (2002)
95. Yi, K., Yu, H., Yang, J., Xia, G., Chen, Y.: Efficient maintenance of materialized top-k views. In: ICDE, pp. 189–200 (2003)
96. Zhang, Y., Lin, X., Xu, J., Korn, F., Wang, W.: Space-efficient relative error order sketch over data streams. In *ICDE*, page 51 (2006)
97. Zhang, Y., Lin, X., Yuan, Y., Kitsuregawa, M., Zhou, X., Yu, J.X.: Summarizing order statistics over data streams with duplicates. In: ICDE, pp. 1329–1333 (2007)
98. Zhu, Y., Shasha, D.: Statstream: Statistical monitoring of thousands of data streams in real time. In: Bressan, S., Chaudhri, A.B., Li Lee, M., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, pp. 358–369. Springer, Heidelberg (2003)