Frugal Streaming for Estimating Quantiles

Qiang Ma[1], S. Muthukrishnan[1], and Mark Sandler[2]

[1] Rutgers University, Piscataway, NJ 08854, USA
{qma,muthu}@cs.rutgers.edu
[2] Google Inc. New York, NY 10011, USA
sandler@google.com

Abstract. Modern applications require processing streams of data for estimating statistical quantities such as quantiles with small amount of memory. In many such applications, in fact, one needs to compute such statistical quantities for each of a large number of groups (*e.g.,*network traffic grouped by source IP address), which additionally restricts the amount of memory available for the stream for any particular group. We address this challenge and introduce *frugal streaming*, that is algorithms that work with tiny – typically, sub-streaming – amount of memory per group.

We design a frugal algorithm that uses *only one* unit of memory per group to compute a quantile for each group. For stochastic streams where data items are drawn from a distribution independently, we analyze and show that the algorithm finds an approximation to the quantile rapidly and remains stably close to it. We also propose an extension of this algorithm that uses *two* units of memory per group. We show experiments with real world data from HTTP trace and Twitter that our frugal algorithms are comparable to existing streaming algorithms for estimating any quantile, but these existing algorithms use far more space per group and are unrealistic in frugal applications; further, the two memory frugal algorithm converges significantly faster than the one memory algorithm.

1 Introduction

Modern applications require processing streams of data for estimating statistical quantities such as quantiles with small amount of memory. A typical application is in IP packet analysis systems such as Gigascope [8] where an example of a query is to find the median packet (or flow) size for IP streams from some given IP addresses. Since IP addresses send millions of packets in reasonable time windows, it is prohibitive to store all packet or flow sizes and estimate the median size. Another application is in social networking sites such as Facebook or Twitter where there are rapid updates from users, and one is interested in median time between successive updates from a user. In yet another example, search engines can model their search traffic and for each search term, want to estimate the median time between successive instances of that search.

Motivated by applications such as these, there has been extensive work in the database community on theory and practice of approximately estimating quantiles of streams with limited memory [1–4, 6, 7, 9–11, 13, 14, 17]. Taken together, this body of research has generated methods for approximating quantiles to $1 + \epsilon$ approximation with space roughly $O(1/\epsilon)$ in various models of data streams.

A. Brodnik et al. (Eds.): Munro Festschrift, LNCS 8066, pp. 77–96, 2013.

Our work here begins with our experience that while the algorithms above are useful, in reality, they get used within GROUPBYs, that is, there are a large number of groups and each group defines a stream within which we need to compute quantiles. In example applications above, this is evident. In IP traffic analysis, one wishes to find median packet size from *each* of the source IP addresses, and therefore the number of "groups" is upto 2^{32} (or 2^{128}). Similarly, in social network application, we wish to compute the median time between updates for *each* user, and the number of users is in 100's of millions for Facebook or Twitter. Likewise, the number of "groups" of interest to search engines is in 100's of millions of search terms. Now, the bottleneck of high speed memory manifests in a different way. We can no longer allocate a lot of memory to any of the groups! In real systems such as Gigascope, low level aggregation engines keep in memory as many groups as they can and rely on higher level aggregation to aggregate partial answers from various groups, which ends up essentially forcing the higher level aggregator to work as a high speed streamer, and proves ineffective.

Motivated by this, we introduce the new direction of *frugal streaming*, that is streaming algorithms that work with tiny amount of memory per group, memory that is far less than is used by typical streaming algorithms. In fact, we will work with 1 or 2 memory locations per group. Our contributions are as follows.

- We present two frugal streaming algorithms for estimating a quantile of a stream. One uses 1 unit of memory for the data stream item, and the other uses 2 units of memory.
- For stochastic streams, that is streams where each item is drawn independently from a distribution, we can mathematically analyze and show how our algorithms converge rapidly to the desired quantile and how they stably oscillate around the quantile as stream progresses.
- We evaluate our algorithms on real datasets from Twitter. We show that our frugal streaming algorithms perform accurately and quickly. Regular streaming algorithms known previously either are highly inadequate given our memory constraints or need significantly more memory to be comparable in accuracy. Further, our frugal algorithms have an intriguing "memoryless" property. Say the stream abruptly changes and now represents a new distribution; *irrespective of the past*, at any given moment, our frugal algorithms move towards the median of the new distribution without waiting for the new streaming items to drown out the old median. We also experimentally evaluate the performance of our frugal streaming algorithms with changing streams.

Ian Munro and Mike Paterson [16], very early on, introduced and solved the problem of estimating quantiles in one or more passes with small memory. This influential paper was a prelude to the area of streaming that was to emerge 15+ years later. Our paper here is an homage to this classical paper.

In Section 2 we introduce definitions and notations. We present our 1 unit of memory frugal streaming algorithm in Section 3. It is analyzed for stochastic streams in Section 4 to give insights about its speed in approaching true quantile and its stability in the long run. Section 5 gives an extension to 2 units of memory frugal streaming algorithm. We discuss related algorithms and present our experimental study in Section 6 and 7. Section 8 has concluding remarks.

2 Background and Notations

Suppose values in domain D are integers[1] distributed over $\{1, 2, 3, \ldots, N\}$. Given a random variable X in domain D, denote its cumulative distribution function (CDF) as $F(x)$, and its quantile function as $Q(x)$. In other words, $F(Q(x)) = x$ if the CDF is strictly monotonic.

h-th p-quantile is x such that $Pr(X < x) = F(x) = \frac{h}{p}$. For convenience we use $\frac{h}{p}$-quantile for the hth p-quantile. S is a sampled set from D. Define a rank function that gives the number of items in S which are smaller than x, $R(x) = |S'|$ where $S' = \{s_i \in S, s_i < x\}$. So when size of S grows to infinity, $F(x) = \frac{R(x)}{|S|}$.

In this paper we consider rank p-quantiles, so the $\frac{h}{p}$-quantile approximation returned by algorithm is considered correct even if the approximation value has zero probability in domain D. For example, if D is distributed over two values 1 and 1000 with equal probabilities. Under value quantile evaluation, a median estimation at 1000 would be considered accurate (throughout our paper, upper median is used for even sample sizes). But any value between 1 and 1000 can also give us good estimation in terms of ranking, hence they are considered correct estimation under rank quantile evaluation.

Throughout when we refer to memory use of algorithms, each memory unit has sufficient bits to store the input domain, that is, each memory unit is $\log N$ bits. This is standard in data stream literature where a method uses f words, it is really f words each of which has sufficient bits to store the input, or $f \log N$ bits.

3 Frugal Streaming Algorithm

We start from median estimation problem and then generalize our algorithm to estimate any quantile of stream S.

3.1 1 Unit Memory Algorithm to Estimate Median

Our algorithm maintains only one unit of memory $\tilde{m}$ which contains its estimate for the stream median, m_S. When a new stream item s_i arrives, consider what our algorithm can do? Since it has no memory of the past beyond $\tilde{m}$, it can do very little. The algorithm adjusts its estimate so that the absolute difference with the new stream item is decreased. C-style pseudo code of this algorithm is described in Algorithm 1, $Frugal$-$1U$-$Median$.

Example 1. To illustrate how $Frugal$-$1U$-$Median$ works, let us consider the example in Figure 1. The estimated median from $Frugal$-$1U$-$Median$ algorithm starts from $\tilde{m}_0 = 0$, and gets updated on each arriving stream item. For example, when $s_4 = 5$ comes, it is larger than $\tilde{m}_3$ whose value is 1, therefore $\tilde{m}_4 = \tilde{m}_3 + 1 = 2$. In this example, $\tilde{m}$ starts from 0, and after reading 5 items from the stream it reaches the stream median for the first time.

[1] For domains with non-integer values, their values can be rewritten to keep desired precision and converted to integers.

Item index i =		1	2	3	4	5	6	7	8	... ...
Stream Items s_i		4	2	1	5	3	2	5	4	... ...
True Medians m_i		4	4	2	4	3	3	3	4	... ...
Median Estimates $\tilde{m}_i$	0	1	2	1	2	3	2	3	4	... ...

Fig. 1. Estimate stream median

Algorithm 1 $Frugal\text{-}1U\text{-}Median$

Input: Data stream S, 1 unit of memory $\tilde{m}$
Output: $\tilde{m}$

```
Initialization m̃ = 0
for each s_i in S do
    if s_i > m̃ then
        m̃ = m̃ + 1;
    else if s_i < m̃ then
        m̃ = m̃ − 1;
    end if
end for
```

3.2 1 Unit of Memory to Estimate Any Quantile

Following the same intuition as above, we can use 1 unit of memory to estimate any $\frac{h}{k}$-quantile, where $1 \leq h \leq k-1$. If the current stream item is larger than estimation, we need to increase estimation by 1; otherwise, we need to decrease estimation by 1. The trick to generalize median estimation to any $\frac{h}{k}$-quantile estimation is that not every stream item seen will cause an update. If the current stream item is larger than estimation, an increment update will be triggered only with probability $\frac{h}{k}$. The rationale behind it is that if we are estimating $\frac{h}{k}$-quantile, and if the current estimate is at stream's true $\frac{h}{k}$-quantile, we will expect to see stream items larger than the current estimate with probability $1-\frac{h}{k}$. If the probability of seeing larger stream items is greater than $1-\frac{h}{k}$, it is caused by the fact that the current estimate is smaller than stream's true $\frac{h}{k}$-quantile. Similarly, a smaller stream item will cause a decrement update only with probability $1-\frac{h}{k}$. Our general 1 unit of memory quantile estimation algorithm is described in Algorithm 2, $Frugal\text{-}1U$.

We need to make a few observations. Besides $\tilde{m}$, this algorithm uses $rand$ and $\frac{h}{k}$. Notice that we can implement the algorithm without explicitly storing $rand$ value, $\frac{h}{k}$ is a constant across all the groups, no matter how many, and can be kept in registers.

Update taken by $\tilde{m}$ in $Frugal\text{-}1U$ is 1, it is a small change at each step when the stream quantile to estimate is large. When it is allowed one extra unit of memory, we can use it to store the size of update to take, denoted as `step`. The extension to two units of memory algorithm is presented in Section 5.

Algorithm 2 $Frugal\text{-}1U$

Input: Data stream S, h, k, 1 unit of memory $\tilde{m}$
Output: $\tilde{m}$

```
Initialization m̃ = 0
for each s_i in S do
    rand = random(0,1); // get a random value in [0,1]
    if s_i > m̃ and rand > 1 − h/k then
        m̃ = m̃ + 1;
    else if s_i < m̃ and rand > h/k then
        m̃ = m̃ − 1;
    end if
end for
```

4 Analysis of $Frugal\text{-}1U\text{-}Median$

Our frugal algorithm for estimating a quantile can behave badly on certain streams. For example, if the true stream quantile value has high probability, even if current estimation is at the true stream quantile, an update of 1 to our estimation will cause large change in rank quantile error. Also any adversary that can remember the entire past and reorder the stream items, they can constantly mislead our algorithm by spreading out the median. This is expected because our algorithm has no memory of the past. The real intuition and strength of our algorithm comes from elsewhere. We say a stream is Stochastic if each stream item is drawn from some distribution D, randomly and independently from other stream items. We will analyze and show that for Stochastic streams, our algorithm quickly converges to an estimate of the target quantile, and further, stably remains in the neighborhood of the quantile as stream progresses.

4.1 Approaching Speed

For our 1 memory algorithm, each update size is 1. At any time t_i, our algorithm estimation has non-zero probabilities to move towards or away from true quantile. Therefore for sufficiently large t, the probability that algorithm estimation moves continuously in one direction is very low. When current algorithm estimation is far away from true quantile, the speed of approaching the true quantile is high, since every update is highly biased towards true quantile. But as the estimation gets closer to true quantile, the bias to move towards true quantile gets weaker so the speed of approaching the true quantile is low. In other words, we are likely to see algorithm estimation showing an oscillating trajectory towards true quantile. The analysis of our algorithm is non-trivial and challenging because the rate of the convergence to an estimate is not constant and depends on a number of varying factors. We rely on the concept of stochastic dominance and we show that in fact the algorithm will approach the true quantile with linear speed.

Recall our notations from Section 2, $F(t)$ is the CDF of distribution, $Q(x)$ is quantile. Let x_i be an indicator variable for the direction of i-th step of the algorithm, where $x_i = 1$ for increment and $x_i = -1$ for decrement. Let $\tilde{m}_t = \sum_{i=1}^{t} x_i$, in other words $\tilde{m}_t$ is the estimation of the quantile at time t. Let $|F(i) - F(i+1)| \leq \delta$, so δ is the

maximum single location probability in distribution and $0 \leq \delta < 1$. Suppose the algorithm is to estimate $\frac{h}{k}$ quantile, whose value is M. Assume the algorithm estimate starts from position $\tilde{m}_0$, where $\tilde{m}_0 < M$. The distance from start position to true quantile is $M - \tilde{m}_0$, but the analysis trivially generalizes to the case where the distance from start position to the true quantile is M.

Lemma 1. *For median estimation, assume the algorithm estimate starts from position $\tilde{m}_0$, where $F(\tilde{m}_0) < \frac{1}{2} - \delta$. After $T = \frac{M|\log 1/\varepsilon|}{\delta}$ steps of algorithm, the probability that $F(\tilde{m}_t) < \frac{1}{2} - \delta$ for all $t < T$ is at most ε. In other words, after $O(M)$ steps it is likely the algorithm has crossed vicinity of the true quantile, $\frac{1}{2} - \delta$, at least once.*

Proof. Let $M' = Q(\frac{1}{2} - \delta)$, we can compute the expectation of a move whenever the algorithm is below M'.

$$\mathbf{Pr}\left[x_i = 1\right] \geq \frac{1}{2}(1 - (\frac{1}{2} - \delta)) = \frac{1}{2} - \frac{1}{2^2} + \delta * \frac{1}{2}$$

we denote it by θ, then

$$\mathbf{Pr}\left[x_i = -1\right] \leq (1 - \frac{1}{2})(\frac{1}{2} - \delta) = \theta - \delta$$

Therefore we have

$$\mathbf{E}\left[x_i\right] \geq \delta \tag{1}$$

In other words the expected shift of each x_i before it hits M' is then at least δ. To prove our lemma, we therefore can use tail inequalities to bound the deviation of $\tilde{m}_t = \sum x_i$ from the expectation. The main difficulty, however arises from the fact x_i are not independent from each other and the constraint (1) holds only when $\tilde{m}_t \leq M'$. Consider an arbitrary sequence of moves x_i. Define $y_i = x_i$ for all $i < i_0$, where i_0 is the time where $\tilde{m}_{i_0}$ crossed M' for the first time, and $y_i = 1$ with probability θ, $y_i = -1$ with probability $\theta - \delta$, and 0 otherwise. Similarly we define $Y_t = \sum y_i$ for all $i < i_0$. Then we have $\mathbf{Pr}\left[\tilde{m}_i < M',\ \forall i \in [T]\right] = \mathbf{Pr}\left[Y_i < M',\ \forall i \in [T]\right]$. Therefore it is enough for us to prove our statement for Y_i. However, Y_i are still not necessarily independent from each other, before they cross M', however all of them satisfy $\mathbf{E}\left[y_i\right] \geq \delta$ and $\mathbf{Pr}\left[y_i = 1\right] \geq \theta$, and $\mathbf{Pr}\left[y_i = -1\right] \leq \theta - \delta$. Define z_i (and Z_i respectively), such that z_i is stochastically dominated by y_i and each z_i is 1 with probability θ and -1 with probability $\theta - \delta$. Using the Hoeffding inequality we have:

$$\mathbf{Pr}\left[|Z_t - \mathbf{E}\left[Z_t\right]| > C\right] \leq \exp(-\frac{tC}{2})$$

using the fact

$$\mathbf{E}\left[Z_t\right] \geq \delta t \geq M|\log 1/\varepsilon| = M - (M \log 1/\varepsilon + M)$$

and using $C = (M + M \log 1/\varepsilon)$ and using union bound over all t we have desired result immediately for Z_t. Using the fact that $Y_t \geq Z_t$ we have the probability that Y_t never crosses the vicinity is less than ε and hence lemma holds.

Note, that our constraints are spelled in terms of probability mass inequality rather than absolute error. This is required, since for any function $f(M)$, it is possible to devise a distribution, such that the algorithm will be $f(M)^2$ far away from true quantile in absolute steps, and yet it will be very close to it in terms of probability mass.

Lemma 2. *For median estimation, algorithm estimation starts from a position $\tilde{m}_0$, where $F(\tilde{m}_0) > \frac{1}{2} + \delta$. After $T = \frac{M|\log 1/\varepsilon|}{\delta}$ steps of algorithm, the probability that $F(\tilde{m}_t) > \frac{1}{2} + \delta$ for all $t < T$ is at most ε.*

Proof. Proof is similar to Lemma 1.

Theorem 1. *For median estimation, algorithm estimation starts from a position $\tilde{m}_0$, where $F(\tilde{m}_0)$ is outside of region $[\frac{1}{2} - \delta, \frac{1}{2} + \delta]$. After $T = \frac{M|\log \varepsilon|}{\delta}$ steps the algorithm, the probability that $F(\tilde{m}_t)$ is outside of this close region $[\frac{1}{2} - \delta, \frac{1}{2} + \delta]$ for all $t < T$ is at most ε.*

Proof. Proof is directly obtained from Lemma 1 and Lemma 2.

In approaching speed analysis, we do not need assumptions on algorithm's starting estimation. Therefore this actually implies for $Frugal$-1U algorithm, quantile estimations adjust to new distribution quantile when the underlying distribution changes, regardless of current estimation position. The speed of approaching new distribution quantile can be determined by Theorem 1. We verified this feature of $Frugal$-1U in experiments on streams with changing distribution, but omit the results in the interest of space.

4.2 Stability

Next we show that after algorithm estimate once reaches true median, the probability of estimate drifting far away from true median is low. Note that Theorem 1 is affecting this estimation drifting process the whole time.

Lemma 3. *To estimate the median, suppose algorithm estimate starts from true median, after t steps the algorithm estimate is at position $F(\tilde{m}_t)$, where*

$$\mathbf{Pr}\left[F(\tilde{m}_t) > \frac{1}{2} + 2\sqrt{\delta \ln \frac{t}{\varepsilon}}\right] \leq \varepsilon.$$

Proof. Define $\omega = 2\sqrt{\delta \ln \frac{t}{\varepsilon}}$. Let us split the interval $[\frac{1}{2}, \frac{1}{2} + \omega]$ into two, $[\frac{1}{2}, \frac{1}{2} + \omega/2]$ and $[\frac{1}{2} + \omega/2, \frac{1}{2} + \omega]$. Our approach is to show that once the algorithm reaches the boundary of the first interval, it is very unlikely to continue through the second interval, without ever dipping back into the first. First of all we note that we need at least $T = \frac{\omega}{\delta}$ more steps of increment than decrement to reach outside of the second interval, and by the way we select the probabilistic weight of the interval, we will need at least $T/2$ to pass through each.

Consider arbitrary outcome of the algorithm where $\tilde{m}_t > T$. Since x changes by at most 1 at every step, there exists j, such that $\tilde{m}_j = \frac{T}{2}$. Therefore the entire space of

events can be decomposed based on the value of j where $\tilde{m}_j = \lfloor T/2 \rfloor$ and for all $i > j$, $\tilde{m}_i > \tilde{m}_j$. Thus:

$$\begin{aligned}\mathbf{Pr}\left[\tilde{m}_t > T\right] &= \sum_{j=0}^{t} \mathbf{Pr}\left[\tilde{m}_t > T, \tilde{m}_i > \tilde{m}_j, \forall i > j\right] \times \mathbf{Pr}\left[\tilde{m}_j = \left\lfloor \tfrac{T}{2} \right\rfloor\right] \\ &\leq \sum_{j=0}^{t} \mathbf{Pr}\left[\tilde{m}_t > T, \tilde{m}_i > \tilde{m}_j, \forall i > j\right]\end{aligned}$$

Let us consider individual term for a fixed j in the sum above. We want to show that each term is at most ε/t. Define $Y_i^{(j)}$ for $i \geq j$, where $Y_i^{(j)} = \tilde{m}_j + \sum_{k=j+1}^{i} y_j$, and $y_i^{(j)} = x_i$ if $X_i' > \tilde{m}_j$, for all $i' < i$, and for the remainder of the segment $y_i^{(j)}$ is random variable that is -1 with probability $p = \frac{1}{2} + \frac{\omega}{2}$ and 1 otherwise. In other words Y_i agrees with $\tilde{m}_i$ until $\tilde{m}_i = \tilde{m}_j$ for the first time after j, after that $Y_i^{(j)}$ becomes independent of $\tilde{m}_i$. We have:

$$\begin{aligned}&\mathbf{Pr}\left[\tilde{m}_t > T, \tilde{m}_i > \tilde{m}_j, \forall i > j\right] \\ &= \mathbf{Pr}\left[Y_t^{(j)} > T, Y_i^{(j)} > Y_j^{(j)}, \forall i > j\right] \\ &\leq \mathbf{Pr}\left[Y_t^{(j)} > T\right]\end{aligned}$$

therefore it is sufficient to compute an upper bound for $\mathbf{Pr}\left[Y_t^{(j)} > T\right]$ for all j. Let Z_i^j be a variable which both stochastically dominates $Y_i^{(j)}$, and is -1 with probability p and 1 otherwise. Since $Y_i^{(j)}$ is -1 with probability of *at least* p, so such variable Z_i^j always exists. Note that Z_i^j are independent from each other for all i, thus we can use standard tail inequality to upper bound $Z_t^{(j)}$, and because of the dominance the result will immediately apply to $Y_i^{(j)}$. Since $Z_i^{(j)}$ only depends on j at the starting point, we can shift it to zero and rewrite out constraint as:

$$\sum_{j=0}^{t} \mathbf{Pr}\left[Z_j > T/2\right] \leq \varepsilon$$

where Z_j is defined as sum $\sum_{i=0}^{j} z_i$, and z_i is -1 with probability p and 1 otherwise. The expected value of Z_j is $(1-p)j - pj = (1-2p)j = -\omega j$. Furthermore by our assumption, $\omega \geq \frac{\delta T}{2}$. Therefore using Hoeffding inequality we have $\mathbf{Pr}\left[Z_j > T/2\right] \leq \exp - \frac{(\omega j + T)^2}{4j}$. Thus it is sufficient for us to show that

$$\exp\left(-\frac{(\omega j + T)^2}{4j}\right) \leq \frac{\varepsilon}{t}, \text{ for all } j < t$$

This constraint is automatically satisfied for all j such that

$$j \geq \frac{4}{\omega^2} \ln \frac{t}{\varepsilon} = j_0.$$

Indeed, if $j > j_0$ we have $(\omega j + T)/4j \geq \frac{\omega^2}{4j} \geq \ln t/\varepsilon$.

On the other hand if $j \leq j_0$, then we have

$$\frac{(\omega j + T)^2}{4j} \geq \frac{T^2\omega^2}{16 \ln t/\varepsilon}$$

but $T \geq \omega/\delta$ and substituting the expression for ω we have:

$$\frac{T^2\omega^2}{4 \ln t/\varepsilon} \geq \frac{\omega^4}{16\delta^2 \ln t/\varepsilon} = \ln t/\varepsilon$$

Thus $\mathbf{Pr}\left[Z_j > T/2\right] \leq \varepsilon/t$, for $j < j_0$, completing the proof.

Lemma 4. *To estimate the median, suppose algorithm estimate starts from true median, after t steps, the algorithm estimate is at position $F(\tilde{m}_t)$, where*

$$\mathbf{Pr}\left[F(\tilde{m}_t) < \frac{1}{2} - 2\sqrt{\delta \ln \frac{t}{\varepsilon}}\right] \leq \varepsilon.$$

Proof. Following the same reasoning in the proof of LEMMA 3, we can prove that the probability of estimation moving far to the left is small. Where we can split the interval $[\frac{1}{2} - \omega, \frac{1}{2}]$ into two $[\frac{1}{2} - \omega, \frac{1}{2} - \omega/2]$ and $[\frac{1}{2} - \omega/2, \frac{1}{2}]$. We can show that once the algorithm reaches the boundary of the first interval, it is very unlikely to continue through the second interval without ever dipping back into the first.

Theorem 2. *To estimate median, after t steps, the probability of the algorithm current position*

$$\mathbf{Pr}\left[\left|F(\tilde{m}_t) - \frac{1}{2}\right| > 2\sqrt{\delta \ln \frac{t}{\varepsilon}}\right] \leq \varepsilon.$$

Proof. This theorem is obtained from Lemma 3 and 4.

These properties of median estimation can be generalized to any quantile $\frac{h}{k}$.

5 Algorithm Extensions

The $Frugal$-$1U$ algorithm described in Section 3 uses 1 unit of memory and is intuitive, and we managed to analyze it; however it has linear convergence to the true quantile. This is effectively by design, because the algorithm does not have the capability to remember anything except the current location. A simple extension to our algorithm is to keep a current step size in memory, and modify it if the new samples are consistently on one side of the current estimate.[2] In this section we describe a 2 units of memory algorithm that we use in experiments for comparison.

Generally the algorithm uses two variables to keep quantile estimate and update size, and one extra bit to keep sign, which indicates the increment or decrement direction of

[2] Another approach that we do not explore here, is to use multiplicative update on step size instead of additive.

Algorithm 3 $Frugal\text{-}2U$

Input: Data stream S, h, k, $\tilde{m}$, step, $sign$
Output: $\tilde{m}$

Initialization $\tilde{m} = 0$, step $= 1$, $sign = 1$
for each s_i in S **do**
  $rand$ = random(0,1);
  if $s_i > \tilde{m}$ **and** $rand > 1 - h/k$ **then**
    step += $(sign > 0)$? f(step) : $-f$(step);
    $\tilde{m}$ += (step> 0) ? $\lceil$step$\rceil$: 1;
    $sign = 1$;
    if $\tilde{m} > s_i$ **then**
      step += $s_i - \tilde{m}$;
      $\tilde{m} = s_i$;
    end if
  else if $s_i < \tilde{m}$ **and** $rand > h/k$ **then**
    step += $(sign < 0)$? f(step) : $-f$(step);
    $\tilde{m}$ - = (step> 0) ? $\lceil$step$\rceil$: 1;
    $sign = -1$;
    if $\tilde{m} < s_i$ **then**
      step += $\tilde{m} - s_i$;
      $\tilde{m} = s_i$;
    end if
  end if
  if $(\tilde{m} - s_i) * sign < 0$ **and** step> 1 **then**
    step$= 1$;
  end if
end for

estimate. Empirically this algorithm has much better convergence and stability property than 1 unit of memory algorithm, however the precise convergence/stability analysis of it is one of our future work. On the intuitive level the algorithm for finding the median works as follows. As before it maintains the current estimate of median but in addition it also maintains an update step that increases or decreases based on the observed values, determined by a function f. More precisely, the step increases if the next element from the stream is on the same side of the current estimate, and decreases otherwise. When estimation is close to true quantiles, step can be decreased to extremely small value.

The increment and decrement factors to be applied to step remains an open problem. step can potentially grow to very large values, so the randomness of the order which stream items appear affects estimation accuracy. For example, if let $\texttt{step}_\texttt{i}$ be the step value at ith update, a multiplicative update of $\texttt{step}_{\texttt{i}+1} = 2 \times \texttt{step}_\texttt{i}$ might be a good choice for a random order stream, which intuitively needs $O(\log M)$ updates to reach true quantile at distance M from current estimate. However in empirical data periodic pattern might be apparent in the stream, for example social network users might have shorter activity intervals at evening, but longer intervals at early morning. Then step can easily get increased to a huge value. It will make the algorithm estimate drift far away from true quantile, hence estimates will have large oscillations.

Therefore to trade off convergence speed for estimation stability we present a version of 2 units of memory algorithm that applies constant factor additive update to step size, where $f(\texttt{step}) = 1$. Full details of the algorithm are described in Algorithm 3. Lines 4-11 handle stream items larger than algorithm estimation, and lines 12-19 handle smaller stream items. For brevity we only look at lines 4-11 in detail. Similar to Algorithm $Frugal$-1U , the key to make $Frugal$-2U able to estimate any quantile is that not every stream item will cause an estimation update, so line 4 enables updates only on "unexpected" larger stream items. `step` is cumulatively updated in line 5. Line 6 ensures minimum update to estimation is 1, and `step` size is only applied in update when it is positive. The reason is that when algorithm estimation is close to true quantile, $Frugal$-2U updates are likely to be triggered by larger and smaller (than estimation) stream items with largely equal chances. Therefore `step` is decreased to a small negative value and it serves as a buffer for value bursts (*e.g.,* a short series of very large values) to stabilize estimations. Lines 8-11 are to ensure estimation do not go beyond empirical value domain when `step` gets increased to very large value. At the end of the algorithm, we reset `step` if its value is larger than 1 and two consecutive updates are not in the same direction. This is to prevent large estimate oscillations if `step` gets accumulated to a large value. This checking is implemented by lines 21-23.

Note that $Frugal$-1U and $Frugal$-2U algorithms are initialized by 0, but in practice they can be initialized by the first stream item to reduce the time needed to converge to true quantiles.

6 Related Work and Algorithms to Compare

There has been extensive work in the database community on theory and practice of approximately estimating quantiles of streams with limited memory (e.g.., [1–4,6,7,9–11, 13, 14, 17]). This body of research has generated methods for approximating quantiles to $1 + \epsilon$ approximation with space roughly $O(1/\epsilon)$ in various models of data streams.

We compare our algorithms with existing algorithms that use constant memory for stochastic streams [11], and also non-constant memory algorithms described in [10,17]. However all the non-constant memory algorithms above use considerably more than 2 persistent variables. While some of the algorithms such as the one described in [1] have a tuning parameter allowing to decrease memory utilization, the algorithm then performs poorly when used with less than 20 variables. Here we briefly overview the algorithms we compare.

6.1 GK Algorithm

Greenwald and Khanna [10] proposed an online algorithm to compute ϵ-approximate quantile summaries with worst-case space requirement of $O(\frac{1}{\epsilon}log(\epsilon N))$. Greenwald-Khanna algorithm (GK) maintains a list of tuples (v_i, g_i, $\triangle_i$), where v_i is a value seen from the stream and tuples are order by v in ascending order. $\sum_{j=1}^{i} g_j$ gives the minimum rank of v_i, and its maximum rank is $\sum_{j=1}^{i} g_j + \triangle_i$. GK is composed of two main operations which are to insert a new tuple in to tuple list when sees a new value, and do compression on the tuple list to achieve the minimum space as possible. Throughout

the updates it is kept invariant that for any tuple we have $\sum_{j=1}^{i} g_j + \triangle_i \leq 2\epsilon N$ to ensure the ϵ-approximate query answers. To make it comparable with our $Frugal$-1U and $Frugal$-2U , we limit the number of tuples maintained by GK . When this memory budget is exceeded we gradually increase ϵ (increment by 0.001) to force compression operation get conducted repeatedly until number of tuples used is within specified budget. In our comparison, we limit the number of tuples to be $t = 20$.

6.2 *q-digest* Algorithm

Tree based stream summary algorithms were studied by Manku et al. [14], Munro and Paterson [16], Alsabti et at. [2], Shrivastava et al. [17] and Huang et al. [12]. In this paper we compare with q-$digest$ algorithm proposed in [17], which is most relevant to our comparison aspects. Their proposed algorithm builds a binary tree on a value domain σ, with depth $\log \sigma$. Each node v in this tree is considered as a bucket representing a value range in the domain, associated with a counter indicating the number of items falling in this bucket. A leaf node represents a single value in domain, and associated with the number of items having this value. Each parent node represents the union of the ranges of children nodes, root node represents the full domain range. This algorithm then keeps merging and removing nodes in the tree to meet memory budget requirement.

For every new stream sample we make a trivial q-$digest$ and merge it with q-$digest$ built so far. Therefore, at any time we can query for a quantile based on the most recently updated q-$digest$. For our evaluation we used number of buckets of $b = 20$ to build tree digests.

6.3 *Selection* Algorithm

Guha and McGregor [11] proposed an algorithm that uses constant memory and operates on random order streams, where the order of elements of the stream have not been chosen by adversary. Their approach is a single pass algorithm that uses constant space and their guarantee is that for a given r (the rank of element of interest) their algorithm returns an element that is within $O(n^{1/2})$ rank of r with probability at least $1 - \delta$. The algorithm does not require prior knowledge of the length of the stream, nor the distribution, which is in common with our $Frugal$-1U and $Frugal$-2U .

This single-pass algorithm ($Selection$) processes the stream in phases, and each phase is composed of three sub-phases namely, *sample*, *estimate* and *update*. Throughout the process, algorithm maintains an interval (a, b) which encloses the true quantile and each phase tries to narrow this interval. At any time algorithm has to keep four variables which are the boundaries a and b, estimation u, and a counter to estimate rank of u. For this algorithm, data size n should be given in order to decide how to divide stream into pieces. By adding one more variable, one can remove this requirement of knowing n beforehand. The proved accuracy guarantee can be achieved when the overall stream is very large. In experiments, to relax the requirement of very large streams we set $\delta = 0.99$, and the version without knowing n in advance is evaluated. [3]

[3] McGregor and Valiant [15] gave a new algorithm using the same space, proving improved approximation with accuracy $n^{1/3+o(1)}$ can be achieved. This algorithm behaves qualitatively similar to the algorithm $Selection$ we have implemented here.

7 Empirical Evaluations

In this section we evaluate our algorithms on both synthetic and two real world data sets. For synthetic data we consider two scenarios, one when data arrive from a static distribution, and one when the distribution changes mid-stream. These tests demonstrate that our algorithms perform well for both scenarios. For real world data we evaluate on HTTP streams [5] and Twitter user tweet streams, where our goals are to evaluate median and 90-% quantile estimates of TCP-flow durations and tweet intervals. As mentioned earlier the structure of our algorithms allow us to estimate quantiles for every stream with 1 or 2 in-memory variables, and the quantile to estimate can be shared by all streams.

Instead of evaluating the absolute error of quantile estimation, we evaluate how far the estimate is from the true quantile by the relative mass error. For example if the estimate of 90-% quantile turned out to be 89-% quantile then the error is 0.01. Throughout our evaluations, we initialize $Frugal\text{-}1U$ and $Frugal\text{-}2U$ algorithm estimates with 0 [4]. For non-constant memory algorithms GK and $q\text{-}digest$, we limit the memory budget to 20 units of their in-memory data structure.

7.1 Synthetic Data

In this section we evaluate algorithms on data streams from a Cauchy distribution (density function $f(x) = \frac{\gamma}{\pi(\gamma^2+(x-x_0)^2}$). The reason we picked Cauchy is because it has a high probability of outliers, the expected value of a Cauchy random variable is infinity, thus we can demonstrate that our algorithms work well in the presence of outliers.

Static Distribution. For our experiments we fix $x_0 = 10000$ and $\gamma = 1250$, and draw 3×10^4 samples. Figure 2 shows the evaluation results. Not only for $Frugal\text{-}1U$ and $Frugal\text{-}2U$, but also most of the algorithms in comparison need some time (some amount of stream items) before getting to a stable quantile estimation. When memory is insufficient for the non-constant memory algorithms, estimation performance degrades much. Due to smaller fixed update size of $Frugal\text{-}1U$, it takes much longer travel than $Frugal\text{-}2U$ to reach stream quantiles.

Dynamic Distribution. Since the algorithms in comparison are not built for estimating changing distributions, we only evaluate $Frugal\text{-}1U$ and $Frugal\text{-}2U$ in the scenario where the underlying distribution of stream changes. We generate three sub-streams drawn from three different Cauchy distributions and feed them one by one to our algorithms. For each of the three sub-streams we sample 2×10^4 items in value domains [10000, 15000], [15000, 20000] and [20000, 25000] respectively.

Figure 3 shows the median and 90-% quantile estimations for $Frugal\text{-}1U$ and $Frugal\text{-}2U$ algorithms. Those sub-streams are ordered by their medians in the sequence of highest, lowest and middle, then they are feed to algorithms one by one. For other algorithms they either need to know the value domain as input or they try to learn upper and lower bounds for the quantile in query, therefore if the stream underlying distribution changes their knowledge about stream are out-dated hence quantile

[4] In practice we can also initialize them with the first stream item.

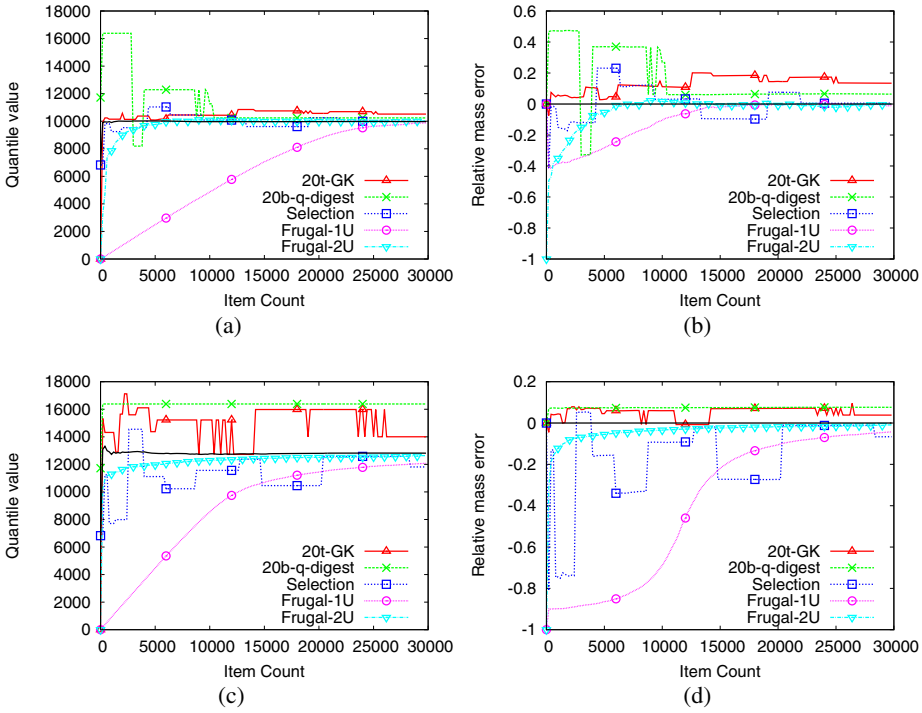

Fig. 2. Evaluation on a stream from one Static Cauchy Distribution. (a) median estimation. (b) relative mass error for (a). (c) 90-% quantile estimation. (d) relative mass error for (c).

approximations are probably not accurate. $Stream\text{-}quantile$ curve shows the cumulative stream quantile, and this is the curve which the other algorithms try to approximate if the combined stream is of interest at the beginning. But in this figure we want to show that our $Frugal\text{-}1U$ and $Frugal\text{-}2U$ are doing a different job. $Use\text{-}Distrib$ curve shows the quantile values for each sub-distribution. The change of $Use\text{-}Distrib$ curve indicates the change of underlying distribution. We can see that our algorithms are trying to reach new distribution's quantile when the stream underlying distribution changes. It is only that $Frugal\text{-}1U$ takes longer time to approach new distribution's quantiles, while $Frugal\text{-}2U$ can make "sharper" turns in its quantile estimations when distribution changes. $Frugal\text{-}1U$ in Figure 3.(b) leaves a steeper approaching trace to 90-% quantile than estimating median in Figure 3.(a), because it is more biased to move estimate towards one direction (getting larger).

One counter argument is that the property of adapting to changing distribution's new quantile also might be a disadvantage, because it makes the algorithms vulnerable to short bursts of "noise". However since the adjustment taken by $Frugal\text{-}1U$ is 1, when the true stream quantile is large the shifting from the true stream quantile caused by short bursts will not affect much in terms of relative mass error. For $Frugal\text{-}2U$ it

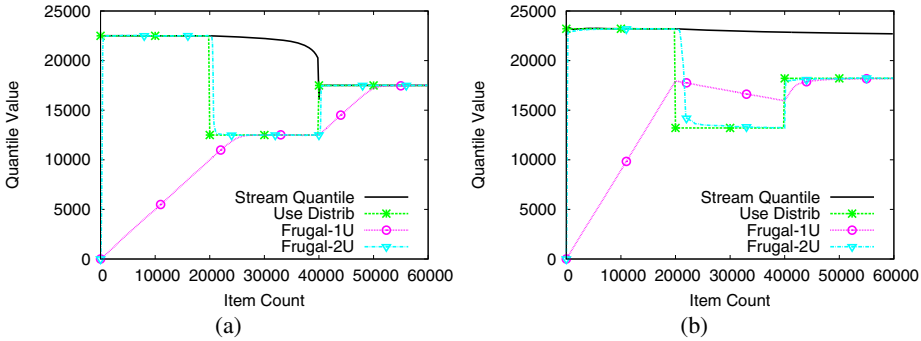

Fig. 3. Evaluation on one stream generated from three Cauchy distributions. (a) Median estimation. (b) 90-% quantile estimation. The change of $Use\text{-}Distrib$ curve indicates the change of underlying distribution. $Frugal\text{-}2U$ algorithm converges to new distribution quantiles significantly faster than $Frugal\text{-}1U$.

is true that `step`'s increment and decrement function f should be picked to trade-off between convergence speed and stability when bursts or periodic patterns are apparent in streams. But once after reaching a close estimate of true quantile, the decreasing `step` value is able to buffer the impact of some value bursts.

7.2 HTTP Streams Data

From an HTTP request and response trace [5] collected for over a period of 6 months, spanning 2003-10 to 2004-03, we extract out TCP-flow durations (in millisecond) between local clients and 100 remote sites, and order them by connections set up time to form streams. In this experiment we first evaluate on streams generated with each of those 100 sites in each of the 6 months. Therefore in total we have 600 streams. But in final performance evaluations we filter out streams with length less than 2000 items and end up with 419 usable streams. Finally we collect the last estimations for median and 90-% quantile by all algorithms.

Figure 4 shows the relative mass error and cumulative percent of 419 TCP-flow duration streams. Figure 4.(a) and (b) show that $Frugal\text{-}2U$ performances are better than or comparable with other algorithms. Whereas $Frugal\text{-}1U$ largely makes underestimations for most of the streams, because in evaluations we initiated $Frugal\text{-}1U$ and $Frugal\text{-}2U$ quantile estimations from 0, however duration stream median (and 90-% quantile) values can easily be tens of thousands. In comparison, $t = 20$ for GK and $b = 20$ for $q\text{-}digest$ are not enough to get close estimations. Note that in relative mass error, the overestimate errors are bounded by 0.5 and 0.1 respectively for median and 90-% quantile estimations.

In the situations where there are millions of streams to be processed simultaneously, statistical quantities about more general groups can help understand the characteristics of different groups. In HTTP request and response trace, streams generated by remote site can also be considered as GROUPBY application to understand the communication patterns from local clients to different remote sites.

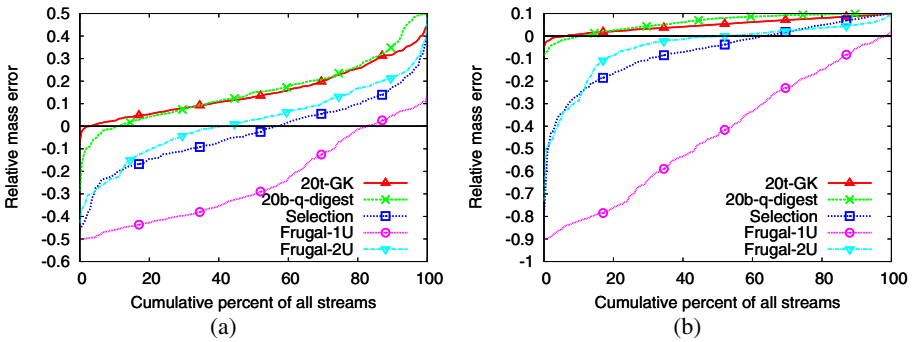

Fig. 4. Evaluation on 419 TCP-flow duration streams by cumulative percent of all streams at different relative mass errors. (a) median estimation. (b) 90-% quantile estimation.

We evaluate all algorithms for one GROUPBY application on this HTTP trace data, where connections with all 100 sites in each month are combined by their creation time. This simulates the viewpoint from trace collecting host. For brevity here we present the results from evaluation on combined stream of month 2004-03, and the results are similar for other months. This combined stream has about 1.6×10^6 items. Figure 5 presents the results on estimating median and 90-% quantile of this stream. In this stream we have median and 90-% quantile values at about 544,267 and 1,464,793 (in microsecond) respectively. Due to the large quantile value $Frugal\text{-}1U$ shows a slower convergence to true stream quantile, while $Frugal\text{-}2U$ handles this problem much better. $Selection$ converges to [-0.1, 0.1] relative mass error region after about 2×10^5 items, but it is oscillatory thereafter and needs much more items to stabilize. In contrast, although $Frugal\text{-}1U$ and $Frugal\text{-}2U$ need relatively more stream items to reach a large true quantile their estimations are relatively stabler. In Figure 5.(a), $b = 20$ $q\text{-}digest$ gives very oscillatory median estimation around 8×10^5, and from the curve it seems converging to stream median but apparently it needs much more stream items. Overall, 20 units of in-memory variables are not sufficient for GK and $q\text{-}digest$ to make accurate quantile estimations.

7.3 Twitter Data Set

From an on-line twitter user directory, we collected 4554 users over 80 directories (e.g. Food and Business). Those tweets from individual users form 4554 sub-streams in the ocean of all tweets. We extracted the intervals (in seconds) between two consecutive tweets for every user and then run our algorithms on those interval streams. This allows us to answer the question of "what is the median inactive time for a given user across all?".

Among the total 4554 twitter users, we filtered out the users with less than 2000 tweets since we need a decent number of data items to reflect the true distribution and allow our algorithms to reach true quantiles. Since twitter does not store more than 3200 tweets of a single user, therefore at the time of data collection the maximum length of a

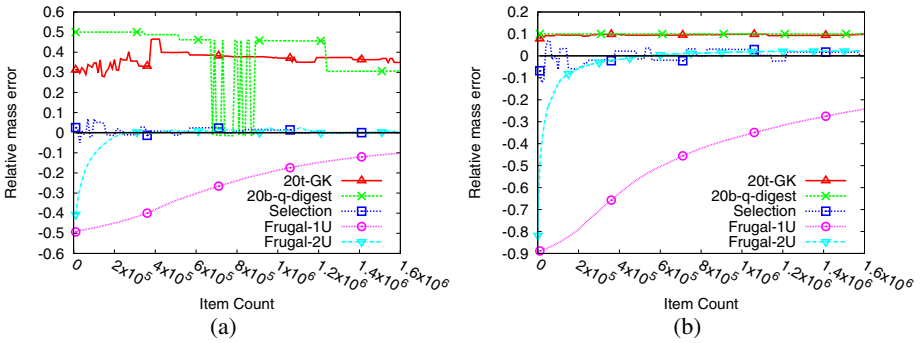

Fig. 5. Relative mass error evaluation on TCP-flow duration stream of month 2004-03. (a) median estimation. (b) 90-% quantile estimation.

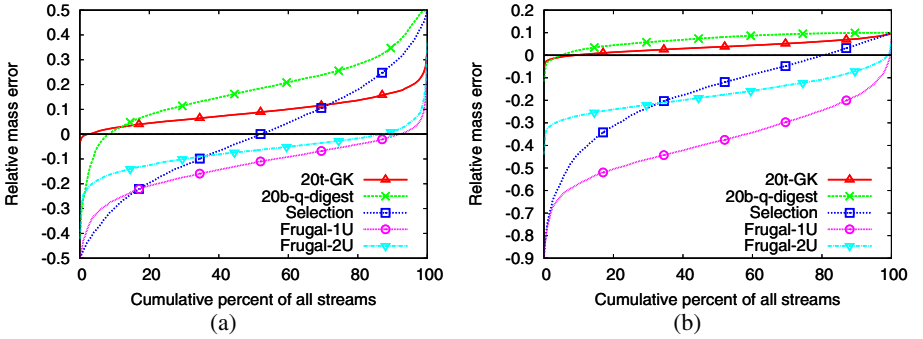

Fig. 6. Evaluation on 4414 twitter users' tweet interval streams, by cumulative percent of all streams at different relative mass errors. (a) median estimation. (b) 90-% quantile estimation.

single user's interval stream is 3200. Finally we evaluated our algorithms on 4414 tweet interval streams, and collected the last estimations for median and 90-% quantile.

Figure 6 shows the relative mass error and cumulative percent of all 4414 interval streams. In Figure 6.(a) we see that about 70 percent of the last median estimation by $Frugal$-1U are under-estimating (less than -0.1). In evaluations we initiated $Frugal$-1U and $Frugal$-2U quantile estimations from 0, however interval stream median (and 90-% quantile) values can easily be tens of thousands. Therefore within 2000 steps they can not fully reach true median. $Frugal$-2U applies dynamic $step$ size hence it performs much better than $Frugal$-1U algorithm, with more than 70 percent of the last median estimations in error range [-0.1, 0.1]. In comparison, $b = 20$ for q-$digest$ are not enough to get close estimations, and $Selection$ does not work well on these short streams. Figure 6.(b) shows that when estimating 90-% quantile, which are much larger values, as expected $Frugal$-1U cannot reach true quantile when the stream items are few (94% of twitter user interval streams have 90-% quantiles larger than 3,200). Again $Frugal$-2U shows its advantages over $Frugal$-1U but it also needs longer streams to reach true quantiles. In comparison, $t = 20$ for GK and $b = 20$ for q-$digest$ are

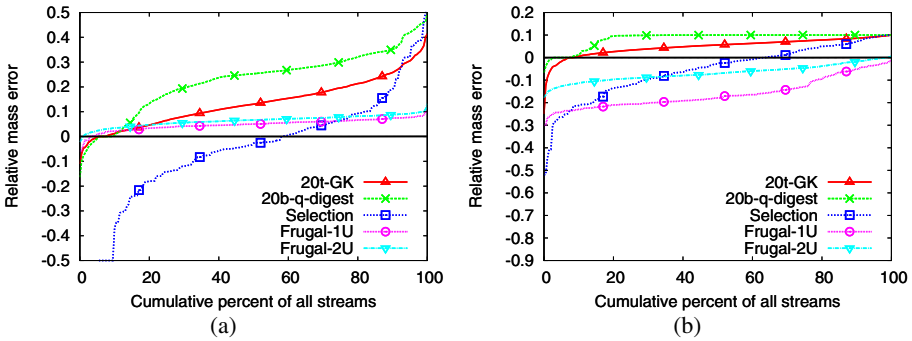

Fig. 7. Evaluation on 905 daily tweet interval streams, by cumulative percent of all streams at different relative mass errors. (a) Median estimation. (b) 90-% quantile estimation.

not affected by stream sizes, however $Selection$ algorithm needs much longer streams. Again note that from this figure, the overestimate errors are bounded by 0.5 and 0.1 respectively for median and 90-% quantile estimations, because relative mass error is measured.

For a database there are various meaningful GROUPBY applications, such as group by geo-location and age for an on-line social network database. To simulate such applications, we evaluate our algorithms on the combined tweet interval streams on each day. We merge tweet interval streams from all 4554 twitter users in our dataset, and sort all the intervals based on the their creation time. We divide the combined interval stream into segments by day, and in total our tweet interval data spans 1328 days from 2008 to 2011. We ran our algorithms on each day's stream and take the last estimations from algorithms to evaluate their accuracy. We filter out the days that have less than 2000 intervals in the daily stream, with similar reason to filter individual tweeter user's tweet interval streams. After filtering process, we have 905 days left. Figure 7 shows the cumulative percent of all days against relative mass error, we can see that median and 90-% quantile under-estimation problems in individual user interval streams are alleviated (in daily interval streams about 67% of the streams have size larger than 3,200). Figure 7.(a) demonstrates that $Frugal$-$1U$ can reach close estimation before the daily interval streams end. $Frugal$-$2U$ does not have much advantage over $Frugal$-$1U$ in these streams, so it just shows similar performance with $Frugal$-$1U$. In Figure 7.(b), for 90-% quantile on most of the days $Frugal$-$1U$ algorithm underestimates the true quantiles by using update size of 1. For median and 90-% quantile estimations by $Frugal$-$2U$ almost all last estimates are in relative mass error range [-0.1, 0.1]. In comparison, $t = 20$ for GK and $b = 20$ for q-$digest$ are not enough to get close estimations, and $Selection$ algorithm needs much more stream items.

Throughout our extensive experiments on synthetic and real-world HTTP trace and twitter data, for streams given enough number of data items in the stream, our 1 and 2 variables stochastic algorithms can achieve quite comparative accuracy against other non-constant and constant memory algorithms, while using much less memory and being very efficient for per item update.

8 Conclusions and Future Directions

We have introduced the concept of frugal streaming and presented algorithms that can estimate arbitrary quantiles using 1 or 2 unit memories. This is very useful when we need to estimate quantiles for each of many groups, as applications demand in reality. These algorithms do not perform well with adversarial streams, but we have mathematically analyzed the 1 unit of memory algorithm and shown fast approach and stability properties for stochastic streams. Our analysis is non-trivial, and we believe it provides a framework for analysis of other statistical estimates with stochastic streams. Further we have reported extensive experiments with our algorithms and several prior quantile algorithms on synthetic data as well as real dataset from HTTP trace and Twitter.

To the best of our knowledge our algorithms are the first that perform well with 2 or less persistent variables per group. In contrast, other regular streaming algorithms, while having other desirable properties, perform poorly when pushed to the extreme on memory consumption like we do with our frugal streaming algorithms.

Our work has initiated frugal streaming, but much remains to be done. First, we need mathematical analysis of 2 or more memory algorithms and at this moment, it looks quite non-trivial. We also need frugal streaming algorithms for other problems such as distinct count estimation and others, that are critical for streaming applications. Finally, as our experiments and insights indicate, frugal streaming algorithms work with so little memory of the past that they are adaptable to changes in the stream characteristics. It will be of great interest to understand this phenomenon better.

References

1. Agrawal, R., Swami, A.: A one-pass space-efficient algorithm for finding quantiles. In: Proc. 7th Intl. Conf. Management of Data, COMAD 1995 (1995)
2. Alsabti, K., Ranka, S., Singh, V.: A one-pass algorithm for accurately estimating quantiles for disk-resident data. In: Proc. 23rd VLDB Conference, pp. 346–355 (1997)
3. Arasu, A., Manku, G.S.: Approximate counts and quantiles over sliding windows. In: Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2004, pp. 286–296. ACM, New York (2004)
4. Babcock, B., Datar, M., Motwani, R., O'Callaghan, L.: Maintaining variance and k-medians over data stream windows. In: Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2003, pp. 234–243. ACM, New York (2003)
5. Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N.: Privacy vulnerabilities in encrypted HTTP streams. In: Danezis, G., Martin, D. (eds.) PET 2005. LNCS, vol. 3856, pp. 1–11. Springer, Heidelberg (2006)
6. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In: Proceedings of the Twenty-Fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2006, pp. 263–272. ACM, New York (2006)
7. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. Journal of Algorithms 55(1), 58–75 (2005)
8. Cranor, C., Johnson, T., Spataschek, O.: Gigascope: a stream database for network applications. In: SIGMOD, pp. 647–651 (2003)

9. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.J.: How to summarize the universe: dynamic maintenance of quantiles. In: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB 2002, pp. 454–465. VLDB Endowment (2002)
10. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. SIGMOD Rec. 30, 58–66 (2001)
11. Guha, S., Mcgregor, A.: Stream order and order statistics: Quantile estimation in random-order streams. SIAM Journal on Computing 38, 2044–2059 (2009)
12. Huang, Z., Wang, L., Yi, K., Liu, Y.: Sampling based algorithms for quantile computation in sensor networks. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, pp. 745–756. ACM, New York (2011)
13. Lin, X., Lu, H., Xu, J., Yu, J.X.: Continuously maintaining quantile summaries of the most recent n elements over a data stream. In: Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, pp. 362–374. IEEE Computer Society, Washington, DC (2004)
14. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: Approximate medians and other quantiles in one pass and with limited memory. SIGMOD Rec. 27, 426–435 (1998)
15. Mcgregor, A., Valiant, P.: The shifting sands algorithm. In: SODA (2012)
16. Munro, J.I., Paterson, M.S.: Selection and sorting with limited storage. Theoretical Computer Science 12(3), 315–323 (1980)
17. Shrivastava, N., Buragohain, C., Agrawal, D., Suri, S.: Medians and beyond: new aggregation techniques for sensor networks. In: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004, pp. 239–249. ACM, New York (2004)