Stratified Reservoir Sampling over Heterogeneous Data Streams

Mohammed Al-Kateb and Byung Suk Lee

Department of Computer Science, The University of Vermont, Burlington VT, USA
{malkateb,bslee}@cs.uvm.edu

Abstract. *Reservoir sampling* is a well-known technique for random sampling over data streams. In many streaming applications, however, an input stream may be naturally *heterogeneous*, i.e., composed of sub-streams whose statistical properties may also vary considerably. For this class of applications, the conventional reservoir sampling technique does not guarantee a statistically sufficient number of tuples from each sub-stream to be included in the reservoir, and this can cause a damage on the statistical quality of the sample. In this paper, we deal with this heterogeneity problem by *stratifying* the reservoir sample among the underlying sub-streams. We particularly consider situations in which the stratified reservoir sample is needed to obtain reliable estimates at the level of either the entire data stream or individual sub-streams. The first challenge in this stratification is to achieve an optimal allocation of a fixed-size reservoir to individual sub-streams. The second challenge is to adaptively adjust the allocation as sub-streams appear in, or disappear from, the input stream and as their statistical properties change over time. We present a stratified reservoir sampling algorithm designed to meet these challenges, and demonstrate through experiments the superior sample quality and the adaptivity of the algorithm.

1 Introduction

Sampling is the process of selecting some members of a population for the purpose of deriving estimates of the population using only the selected members [10] [14]. The basic sampling scheme is *random sampling* in which each member of the population has an identical chance of being in the sample. Random sampling usually generates consistent and unbiased estimates of the original population, and it has been used in a wide range of application domains such as approximate query processing (e.g., [23]) and data stream processing (e.g., [20]).

For applications in which data are available in the form of an incoming *stream*, sampling has two major challenges. First, the size of the data stream is usually unknown a priori and, therefore, it is not possible to predetermine the sample fraction (or sampling rate) before the sampling starts. Second, in most cases the data arriving in a stream cannot be stored and, therefore, have to be processed sequentially in a single pass. A technique commonly used to overcome these challenges is the *reservoir sampling* [15] [22], which selects a random sample of a fixed size without replacement from a stream of an unknown size.

Table 1. Mean and standard deviation of bidding amount of two FCC auctions [2]

Auction Item	**Mean**	**Std dev**
FCC 700 MHz Guard Band	73964.29	24591.07
FCC Analog TV Stations	263800.00	39115.21

For many streaming applications, however, we make two key observations. First, an input stream may be composed of *sub-streams* that correspond to different groups whose statistical properties, specifically *mean* and *variance*, may vary significantly. We refer to this class of data streams as *heterogeneous* data streams. Second, the application may naturally demand using a sample to derive estimates at the level of either the entire data stream or individual sub-streams.

Consider, for example, the application of the Federal Communications Commission (FCC) auction system [2] through which auctions for licenses of electromagnetic spectrum are conducted electronically over the Internet. In this application, an auction data stream is composed of *multiple sub-streams* each of which represents the biddings in a particular auction. Moreover, an auction data stream can be *heterogeneous*, as the mean and the variance of bidding amounts vary significantly from one auction to another depending on the type of an auction item (see Table 1). From the application standpoint, the scope of sampling differs in two ways. On one hand, a sample of *all* bidding amounts can be used to perform a set of analyses on the entire auction data stream, e.g., the median of bidding amounts across all auctions. On the other hand, a sample of the bidding amounts in *individual* auctions can be utilized to generate the estimate for each individual auction, e.g., the median of bidding amounts in each auction.

For such applications with heterogeneous data streams, the conventional reservoir sampling technique does not guarantee a statistically sufficient number of tuples to be included in the reservoir for *every* sub-stream. The inevitable consequence of this is a damage to the statistical quality of the sample. Furthermore, the technique is only used for the purpose of maintaining one random sample of a fixed size from all tuples seen so far in an input stream. Therefore, it is not suitable when multiple random sub-samples are needed to obtain the estimates of individual sub-streams. In other words, it is not appropriate for the purpose of producing a *sub-sample* stored in a *sub-reservoir* for each *sub-stream.*

The research literature addresses an analogous heterogeneity problem in the context of database systems through *stratified sampling* [9] [13]. In this sampling scheme [10], a population is initially clustered into homogenous disjoint strata. Then, a sample is taken randomly from each stratum. Stratified sampling is particularly preferred if the statistical properties of strata vary considerably [14]. Statistical properties are typically mean and variance or, equivalently, *coefficient of variation (CV)* which is the ratio of the standard deviation to the mean.

In no existing work, however, *data stream* has been the target of a stratified sampling algorithm. When applied to data streams, stratified sampling inherits the challenges of random sampling over a data stream and poses the following additional challenges. First, usually neither the number of sub-streams nor their statistical properties are known in advance. Thus, it is not possible to optimally allocate a stratified sample to sub-streams prior to sampling. Second, the membership of a data stream and the statistical properties of the member sub-streams may change over time. Hence, the allocation should have the ability to adapt to these changes.

In this paper, we address the problem of maintaining a stratified reservoir sample over heterogeneous data streams for applications that demand reliable estimates at the level of either the entire data stream or individual sub-streams. There are two specific problems to be solved. The first one is to allocate a given fixed-size reservoir optimally among sub-streams, and the second one is to adjust the allocation as new sub-streams appear or existing sub-streams disappear (e.g., due to punctuation) or their statistical properties change over time.

To solve the allocation problem, we adopt a statistical method known as the *power allocation* [6]. By controlling what is called the *power* parameter, this method allows to allocate a given sample size optimally [6] when the estimates are required from the data stream or from the individual sub-streams. To adapt to changes in data stream membership and sub-streams statistical properties, *uniformity* of the sample of each sub-stream should be maintained as the corresponding sample size changes over time. For this we use a simple variation of the *adaptive-size reservoir sampling* technique (from our prior work) [5], which maintains the uniformity of a reservoir sample with a required degree of confidence after the reservoir size is adjusted in the middle of sampling.

Two sets of experiments have been conducted using synthetic and real datasets. In the first set of experiments, we compare the stratified and the conventional reservoir sampling algorithms with respect to the sample quality – specifically, sample *accuracy* and sample *precision*[1] – for different number of input sub-streams and for varying degree of heterogeneity among the sub-streams. The results of this experiment show that the stratified algorithm outperforms the conventional algorithm by nearly an order of magnitude in both sample quality metrics. In the second set of experiments, we examine how adaptively the stratified reservoir sampling algorithm adjusts the allocation of the fixed reservoir sample size. The results of this experiment show the stratified reservoir sampling quickly adjusting the sub-sample sizes when the CVs of the member sub-streams change and when a new sub-stream appears or an exiting sub-stream expires.

Main contributions of this paper can be summarized as follows.

– It identifies and motivates the problem of stratified reservoir sampling over heterogeneous data streams.
– It presents an algorithm for maintaining a stratified reservoir sample over a heterogeneous data stream when the sample is used to obtain either one estimate from the whole stream or multiple estimates from the sub-streams.
– It empirically shows the superiority of the proposed algorithm (with respect to the precision and accuracy of the sample) and demonstrates its adaptivity.

[1] Sample accuracy is the degree of closeness of the estimate to its true value. Sample precision is the degree to which the estimates from different samples taken from the same data set vary from one another.

The rest of the paper is organized as follows. Section 2 gives an overview of the reservoir sampling and stratified sampling techniques. Section 3 formulates the research problem and presents the proposed stratified sampling algorithm. Section 4 presents and discusses the experiment results. Section 5 reviews related work. Section 6 concludes this paper and suggests future work.

2 Background

This section provides backgrounds on reservoir sampling and stratified sampling.

2.1 Reservoir Sampling

Reservoir sampling [15] [22] is a technique for selecting a uniform random sample of a fixed size without replacement from an input stream of an unknown size. Initially, the algorithm (see Algorithm 1) places all tuples in a reservoir r until the reservoir (of the size of $|r|$ tuples) becomes full. After that, each k^{th} tuple is accepted for inclusion in the reservoir with the probability of $\frac{|r|}{k}$ and an accepted tuple replaces a randomly selected tuple in the reservoir.

Algorithm 1. *Conventional Reservoir Sampling* (CRS)

```
Require: $|r|$ // size of a reservoir $r$
1: $k = 0$
2: for each tuple arriving from the input stream do
3:     $k = k + 1$
4:     if $k \leq |r|$ then
5:         add the tuple to the reservoir
6:     else
7:         decide with the probability $\frac{|r|}{k}$ whether to accept the tuple
8:         if the tuples is accepted then
9:             replace a randomly selected tuple in the reservoir with the accepted tuple
10:        end if
11:    end if
12: end for
```

Reservoir sampling guarantees that a reservoir always holds a *uniform* sample of the k tuples seen so far [15]. After the k^{th} tuple arrives, each of the k tuples has the equal probability $\frac{|r|}{k}$ to be included in the reservoir. That is, each of the $\binom{k}{|r|}$ different possible samples has the same probability $\frac{1}{\binom{k}{|r|}}$ to represent r.

2.2 Stratified Sampling

Stratified sampling [10] [14] is a sampling scheme in which a heterogeneous population R is initially clustered into n disjoint homogeneous strata, R_1, R_2, ..., R_i, ..., R_n, and then a sample r_i is taken randomly from each stratum R_i. Every member of R should belong to one and only one stratum (i.e., $R_i \cap R_j = \phi$ $(i \neq j)$ and $R_1 \cup R_2 \cup ... \cup R_i \cup ... \cup R_n = R$). A stratified sample of a given size is expected to have higher statistical precision (i.e., lower sampling error) than a random sample of the same size taken from the same population when the statistical properties (i.e., mean and variance) of strata vary considerably.

Allocating a given sample size $|r|$ to different strata is a fundamental issue in stratified sampling. Obviously, the allocation is under the constraint on the the sum of the sub-sample sizes assigned to individual strata, $|r_1|, |r_2|, ..., |r_n|$:

$$\sum_{i=1}^{n} |r_i| \leq |r| \tag{1}$$

There are two allocation methods commonly used for a stratified sample, the *proportional* allocation [14] and the *Neyman* allocation [14]. Under the *proportional* allocation, the sample size of each stratum is determined in proportion to the size of the stratum:

$$|r_i| = |r| \times \frac{|R_i|}{|R|} \tag{2}$$

where R denotes the whole population, R_i denotes a stratum, and $|R|$ and $|R_i|$ denote the sizes of R and R_i, respectively. Under the *Neyman* allocation, the sample size of each stratum is determined in proportion to the standard deviation as well as the size of the stratum:

$$|r_i| = |r| \times \frac{\sigma_i \times |R_i|}{\sum_{j=1}^{n} \sigma_j \times |R_j|} \tag{3}$$

where σ_i denotes the standard deviation of R_i.

3 Stratified Reservoir Sampling

The proposed stratified reservoir sampling algorithm is described in this section. As mentioned in Section 1, there are two technical issues to resolve in the proposed algorithm: (1) determining the optimal sizes of sub-samples for each sub-stream, and (2) maintaining the uniformity of sub-samples as their sizes change. In this section, we first formulate the problem formally in Section 3.1 and discuss our approaches to the two technical issues in Sections 3.2 and 3.3 and then summarize them into one algorithm in Section 3.4.

3.1 Problem Formulation

The problem of allocating a fixed-size reservoir to sub-streams is an adaptive optimization problem formulated as follows. An input data stream S consists of n sub-streams S_1, S_2, ..., S_n. Each sub-stream S_i $(i = 1, 2, ..., n)$ is a sequence of tuples $s_{i_1}, s_{i_2}, ...$ such that $S_i \cap S_j = \phi$ $(i \neq j)$ and $\cup_{S_i} = S$. Given a total available size of $|r|$ tuples in a reservoir r, the objective is to allocate $|r|$ *optimally* among the n sub-streams subject to the following constraint at any point in time t:

$$\sum_{i=1}^{n} |r_i(t)| \leq |r| \tag{4}$$

where $r_i(t)$ denotes the sample allocated for S_i at time point t and $|r_i(t)|$ denotes its size. The optimality criterion is the sample quality, and there is some minor

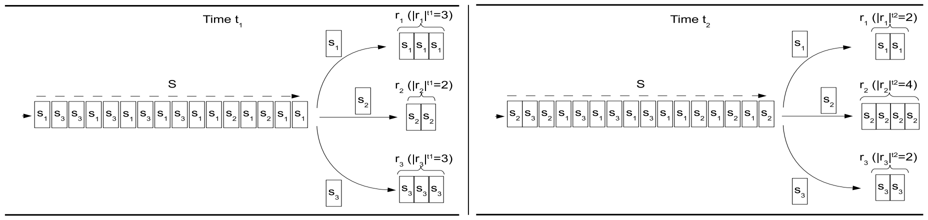


Fig. 1. An illustration of stratified reservoir sampling

difference in the specific criteria depending on which purpose (i.e., one whole sample or individual sub-samples) the sample is used for (details in Section 3.2).

Figure 1 illustrates the processing of stratified reservoir sampling. It shows that the sizes of sub-samples r_1, r_2, and r_3 have respectively decreased, increased, and decreased from t_1 to t_2 while the same total sample size remains the same.

3.2 Optimal Stratified-Reservoir Allocation

For the flexible aim of generating estimates from the whole sample or from separate sub-samples, the commonly used *Neyman allocation* is not adequate enough since it is geared for the former case only. To overcome this limit, we adopt another statistical method, known as *power allocation*, [6]. The power allocation method provides a way to allocate the sample to different strata whether the sample is used to generate a single estimate for the underlying population as a whole or multiple estimates separately for each of the underlying strata. This flexibility is enabled by a control parameter called the *power* of allocation.

Formally, the size of a sample, $|r_i|$, assigned to a stratum R_i is computed as

$$|r_i| = |r| \times \frac{\sigma_i \times \left(\left(\sum_{j=1}^{|R_i|} y_{ij}\right)^q\right) / \left(\frac{\sum_{j=1}^{|R_i|} y_{ij}}{|R_i|}\right)}{\sum_{k=1}^{n} \sigma_k \times \left(\left(\sum_{j=1}^{|R_k|} y_{kj}\right)^q\right) / \left(\frac{\sum_{j=1}^{|R_k|} y_{kj}}{|R_k|}\right)} \tag{5}$$

where y_{ij} denotes the sampling attribute value of the j^{th} member in R_i, σ_i denotes the standard deviation of the sampling attribute values in R_i, and q denotes the power of allocation.

When the stratified sample is used of the *entire population*, power allocation aims to minimize the sampling variance of the estimator of the whole stratified sample, where the sampling variance is formulated as

$$\sum_{i=1}^{n} \sigma_i \times \frac{|R_i| \times (|R_i| - |r_i|)}{|r_i|} \tag{6}$$

In this case, it achieves an optimal allocation by setting the power value to 1. Note that this results in the exact Neyman allocation, that is

$$|r_i| = |r| \times \frac{\sigma_i \times \left(\left(\sum_{j=1}^{|R_i|} y_{ij}\right)^1\right) / \left(\frac{\sum_{j=1}^{|R_i|} y_{ij}}{|R_i|}\right)}{\sum_{k=1}^{n} \sigma_k \times \left(\left(\sum_{j=1}^{|R_k|} y_{kj}\right)^1\right) / \left(\frac{\sum_{j=1}^{|R_k|} y_{kj}}{|R_k|}\right)} = |r| \times \frac{\sigma_i \times |R_i|}{\sum_{k=1}^{n} \sigma_k \times |R_k|} \tag{7}$$

When the stratified sample is used at the level of *individual strata*, Neyman allocation may cause the sampling variances of some strata to be larger than those achievable by considering strata individually. Power allocation's remedy for this deficiency is to allocate sub-sample sizes in proportion to CV of each stratum, which is achieved by setting the power to 0. In this case,

$$|r_i| = |r| \times \frac{\sigma_i \times \left(\left(\sum_{j=1}^{|R_i|} y_{ij}\right)^0\right) / \left(\frac{\sum_{j=1}^{|R_i|} y_{ij}}{|R_i|}\right)}{\sum_{k=1}^{n} \sigma_k \times \left(\left(\sum_{j=1}^{|R_k|} y_{kj}\right)^0\right) / \left(\frac{\sum_{j=1}^{|R_k|} y_{kj}}{|R_k|}\right)} = |r| \times \frac{\sigma_i / \left(\frac{\sum_{j=1}^{|R_i|} y_{ij}}{|R_i|}\right)}{\sum_{k=1}^{n} \sigma_k / \left(\frac{\sum_{j=1}^{|R_k|} y_{kj}}{|R_k|}\right)} \tag{8}$$

Applying this power allocation to data stream gives the following formula for determining sub-sample sizes at any point in time t.

$$|r_i(t)| = |r| \times \frac{\sigma_i(t) \times \left(\left(\sum_{j=1}^{|S_i(t)|} y_{ij}\right)^q\right) / \left(\frac{\sum_{j=1}^{|S_i(t)|} y_{ij}}{|S_i(t)|}\right)}{\sum_{k=1}^{n} \sigma_k(t) \times \left(\left(\sum_{j=1}^{|S_k(t)|} y_{kj}\right)^q\right) / \left(\frac{\sum_{j=1}^{|S_k(t)|} y_{kj}}{|S_k(t)|}\right)} \tag{9}$$

where $|r_i(t)|$ denotes the size of a sub-sample allocated for a sub-stream S_i at time point t, $\sigma_i(t)$ denotes the running standard deviation[2] of the sampling attribute values in S_i up to t, and $|S_i(t)|$ denotes the number of tuples processed up to t from S_i.

3.3 Maintaining Sample Uniformity

As mentioned in Section 1, we use the *adaptive-size reservoir sampling* algorithm (ARS) [5] (see Algorithm 2) to maintain the uniformity of each sub-sample as its size decreases or increases as a result of optimal allocation.

ARS is based on the concept of uniformity confidence (UC), which refers to the probability that a sampling algorithm generates a uniform random sample after the sample size changes in the middle of sampling. A theoretical study in [5] concludes that if the reservoir size decreases, the sample uniformity can be maintained in the *reduced* reservoir with 100% confidence by randomly evicting tuples from the original reservoir.

In contrast, if the reservoir size increases, it is not possible to attain 100% confidence in the *enlarged* reservoir. It is possible, however, to ensure the uniformity confidence above a given threshold. The steps are as follows. First, ARS finds the minimum number of incoming tuples that should be considered to refill the enlarged reservoir such that the resulting uniformity confidence exceeds the given threshold (Equation 10). Then, it decides probabilistically on the number of tuples to retain in the enlarged reservoir and randomly evicts the remaining

[2] Running standard deviation is required for the calculations in the power allocation method. For this, we use an efficient recurrence relation [21], known as the *updating* method, that is capable of calculating the standard deviation in a single scan of the data and providing precise calculation even when the data values are relatively large.

number of tuples (Equation 11). Eventually, it fills the room available in the enlarged reservoir from the incoming tuples.

$$UC(k, |r|, \delta, m) = \frac{\sum_{x=\max\{0,(|r|+\delta)-m\}}^{|r|} \binom{k}{x}\binom{m}{|r|+\delta-x}}{\binom{k+m}{|r|+\delta}} \times 100 \tag{10}$$

$$p(x) = \frac{\binom{k}{x}\binom{m}{|r|+\delta-x}}{\binom{k+m}{|r|+\delta}} \tag{11}$$

In this paper, we use a simple variation of ARS in which the number of incoming tuples required to refill an enlarged reservoir is computed as[3]

$$m = \frac{\delta \times k}{|r|} \tag{12}$$

Algorithm 2. *Adaptive-size Reservoir Sampling* (ARS)

```
Require: $|r|$ // size of a reservoir r
         $k$ // number of tuples seen so far
         $\zeta$ // uniformity confidence threshold
 1: while true do
 2:     while the reservoir size $|r|$ does not change do
 3:         continue sampling using CRS (Algorithm 1)
 4:     end while
 5:     if reservoir size is decreased by $\delta$ then
 6:         randomly evicts $\delta$ tuples from the reservoir
 7:     else
 8:         // i.e., reservoir size is increased by $\delta$
 9:         find the minimum value of $m$ (Equation 10) such that $UC >= \zeta$
10:         flip a biased coin to decide on $x$, the number of tuples to retain in the reservoir (Equation 11)
11:         randomly evict $|r| - x$ tuples from the reservoir
12:         select $\delta + |r| - x$ tuples from the incoming $m$ tuples using CRS (Algorithm 1)
13:     end if
14: end while
```

3.4 Stratified Reservoir Sampling Algorithm

Based on the discussions above, our stratified reservoir sampling algorithm works as shown in Algorithm 3. In this algorithm, the input stream S is treated as a *set* of sub-streams S_1, S_2, etc, and ALG_i refers to the sampling algorithm currently in use for the sub-stream S_i.

In the initialization phase of the algorithm (Lines 1-15), the first $|r|$ tuples in a data stream S are added to the reservoir while the running statistics of sub-streams are being updated (Lines 3-4). The sampling starts using CRS for all new sub-streams (Lines 5-8) and, once the reservoir becomes full, the size of a sub-reservoir is initialized in proportion to the number of tuple seen so far from the corresponding sub-stream (Lines 13-15).

In the sampling phase (Lines 16-41), each time a new tuple s arrives from a sub-stream S_i, the algorithm decides to sample s using CRS if S_i is a new

[3] The rationale for computing the value of m in this way is a simple heuristic that, since r has been filled from k tuples so far, the room for additional δ tuples should be filled in proportion to $\frac{k}{|r|}$, that is, $\delta \times \frac{k}{|r|}$ tuples. This heuristic facilitates the use of the ARS by eliminating the need to conduct an expensive search to find the optimum value of m using Equation 10, which is an inverse-mapping problem.

Algorithm 3. Stratified Reservoir Sampling *(SRS)*

```
Require: $|r|$ // size of a reservoir $r$
         $q$ // power of allocation
         $\Delta$ // sample reallocation time interval
   // ***************INITIALIZATION PHASE***************
 1: for each new tuple $s$ arriving from a sub-stream $S_i$ do
 2:    if reservoir $r$ is not full then
 3:       add $s$ to $r$
 4:       update the running statistics of $S_i$
 5:       if $S_i \notin S$ // i.e., $S_i$ is a new sub-stream then
 6:          $S = S\ U\ \{S_i\}$
 7:          $ALG_i$ = CRS // start sampling using CRS
 8:       end if
 9:    else
10:       break // go to line 13
11:    end if
12: end for
13: for each $S_i \in S$ do
14:    $|r_i|$ = size($S_i$) // initialize sub-reservoir sizes
15: end for
   // ****************SAMPLING PHASE****************
16: while true do
17:    for each new tuple $s$ arriving from a sub-stream $S_i$ do
18:       if $S_i \notin S$ // i.e., $S_i$ is a new sub-stream then
19:          $S = S\ U\ \{S_i\}$
20:          $ALG_i$ = CRS // start sampling using CRS
21:       end if
22:       sample $s$ into the sub-reservoir $r_i$ using $ALG_i$ // either CRS or ARS
23:       update the running statistics of $S_i$
24:       if the time interval $\Delta$ has passed then
25:          break // go to line 28 to calculate sub-reservoir sizes
26:       end if
27:    end for
28:    for each sample $r_i$ allocated to sub-stream $S_i$ do
29:       if $S_i$ expires from $S$ // e.g., due to a punctuation then
30:          S = S - $\{S_i\}$
31:          $|r_i(t)| = 0$
32:       else
33:          calculate $|r_i(t)|$ for $S_i$ using Equation 9 with the given value of $q$
34:          if $|r_i(t)|$ has changed as a result then
35:             $ALG_i$ = ARS (Algorithm 2)
36:          else
37:             $ALG_i$ = CRS (Algorithm 1)
38:          end if
39:       end if
40:    end for
41: end while
```

sub-stream (Lines 18-21). Then, the algorithm samples s into r_i using the corresponding sampling algorithm (i.e., either CRS or ARS) while updating its running statistics (Lines 22-23 and 28-40). Periodically, the algorithm reallocates the reservoir size optimally among sub-streams (Lines 24-26). Specifically, if a sub-stream has expired from the input stream (e.g., due to the presence of a punctuation), the memory of the sub-reservoir of that sub-stream is released (Lines 29-31). Otherwise, the algorithm calculates the optimal sample size for the sub-stream (Line 33). If the size of r_i changes as a result, then the algorithm switches over to ARS to continue sampling the incoming S_i tuples (Lines 34-35). Note that ARS quickly resumes CRS once the size adjustment is handled. If the size of r_i does not change, then the algorithm samples the incoming S_i tuples using CRS (Line 37).

4 Performance Evaluation

We conduct two sets of experiments. The first set of experiments evaluates the performance of the *stratified reservoir sampling (SRS)* algorithm against the *conventional reservoir sampling (CRS)* algorithm with respect to the sample quality. The second set demonstrates the adaptivity of the SRS to the changes of data stream membership and the statistical characteristics of member sub-streams. In this section, the design and setup of the experiments are described in Section 4.1 and the results of the experiments are presented in Section 4.2.

4.1 Experiment Design and Setup

Intuitively, two factors affect the performances of algorithms over a data stream consisting of multiple heterogeneous sub-streams: the number of sub-streams and the degree of heterogeneity among the sub-streams. These two parameters are thus used in the comparisons between SRS and CRS.

Performance Metrics. The two kinds of sample quality mentioned in Section 1 are used to compare the performances of SRS and CRS: *accuracy* and *precision*. Specifically, we use the *error in estimated mean (EEM)*, the difference between the mean value estimated from the sample and the actual mean value, as the metric of sample accuracy. The estimated mean for a random sample is calculated as

$$\frac{1}{|r|} \times \sum_{i=1}^{|r|} y_i \tag{13}$$

where y_i denotes the value of the sampling attribute of the i^{th} tuple in a sample r [10]. Extended from it, the estimated mean for a stratified sample is calculated as

$$\sum_{i=1}^{|n|} \left(\frac{|S_i|}{|S|} \times \left(\frac{1}{|r_i|} \sum_{j=1}^{|r_i|} y_{ij} \right) \right) \tag{14}$$

where y_{ij} denotes the value of sampling attribute of the j^{th} tuple in a sub-sample r_i [10].

On the other hand, we use the *standard error (SE)*, a common statistical quantification of the sample precision, as the metric of sample precision. The SE is a measure of how precise the sample is; the larger the SE, the lower the statistical precision of the sample is, and vice versa. The SE for a random sample is computed as

$$\sqrt{\left(\left(1 - \left(\frac{|r|}{|S|} \right) \right) \times \left(\frac{\sigma^2}{|S|} \right) \right)} \tag{15}$$

where σ^2 denotes the variance of the entire sample [10]. Extended from it, the SE for a stratified sample is computed as

$$\frac{1}{|S|} \times \sqrt{\sum_{i=1}^{n} |S_i|^2 \times \left(1 - \frac{|r_i|}{|S_i|} \right) \times \left(\frac{{\sigma_i}^2}{|r_i|} \right)} \tag{16}$$

where ${\sigma_i}^2$ denotes the variance of the i^{th} sub-sample [10].

Datasets. Experiments are conducted using both synthetic and real datasets. Synthetic datasets are used to examine the effect of the statistical characteristics of an input data stream on the quality of the sample. Now, we describe the process of synthetic dataset generation and outline the profile of the real datasets.

Synthetic Datasets. A synthetic data stream is generated bottom up, that is, by first generating sub-streams and then combining them to form one stream. The sampling attribute value in each sub-stream S_i has the *doubly-truncated normal distribution* [19], i.e., the normal distribution with bounded lower and upper ends. Formally, if a random variable $X \backsim N(\mu, \sigma)$ has the normal distribution such that $\infty \leq l \leq X \leq u \leq \infty$, then X is considered to have a doubly-truncated normal distribution with the probability density function

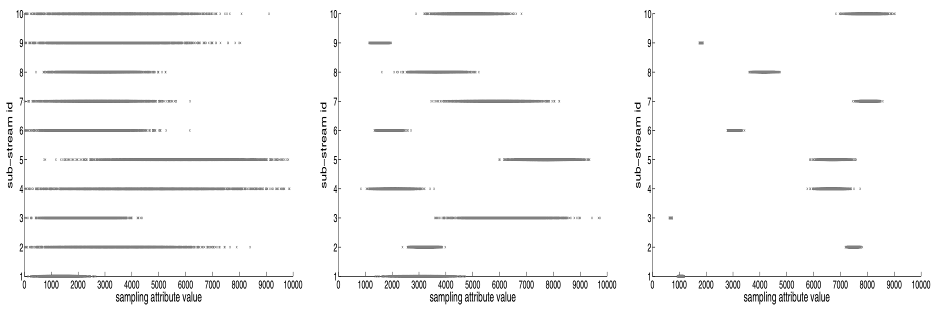
$$pdf\,(x; \mu, \sigma, l, u) = \frac{\frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{u-\mu}{\sigma}\right) - \Phi\left(\frac{l-\mu}{\sigma}\right)} \tag{17}$$

where $\phi(x)$ is the probability density function of the standard normal distribution, and $\Phi(x)$ is its cumulative distribution function [19]. This distribution is used in many applications like inventory management and financial applications, in which the values are naturally constrained within a certain bound [16].

The datasets are synthesized from a different number of sub-streams (n) and with a varying degree of heterogeneity among the sub-streams (*DH*). *DH* is defined as the ratio of the *inter-sub-stream* variability to the *intra-sub-stream* variability. With the variability expressed in terms of CV [6], we define *DH* as the ratio of the *standard deviation* among the CVs of sub-streams ($\sigma_{[CV]}$) to the *average* of the CVs of sub-streams ($\mu_{[CV]}$). With the doubly-truncated normal distribution in place, we know that the standard deviation of the sampling attribute values of a sub-stream S_i is bounded by half the range of these values. This means that each CV_i is bounded within the range of 0 to 1. Consequently, *DH* is also bounded within the range of 0 to 1.

Given the values of n and *DH*, the synthetic dataset generator works as follows. First, it sets the value of $\mu_{[CV]}$ to 0.5 (note $0 < \mu_{[CV]} \leq 1$) and calculates the value of $\sigma_{[CV]}$ accordingly. Second, it generates n random numbers from a doubly-truncated normal distribution with $\mu_{[CV]}$, $\sigma_{[CV]}$, $l_{[CV]} = \mu_{[CV]} - \sigma_{[CV]}$, and $u_{[CV]} = \mu_{[CV]} + \sigma_{[CV]}$. The n random numbers generated correspond to the CVs of the n sub-streams. Third, for S_i, the synthetic dataset generator uses the value of CV_i to assign the values of $\mu_{[S_i]}$ and $\sigma_{[S_i]}$ randomly such that $\frac{\sigma[S_i]}{\mu_{[S_i]}} = CV_i$. Finally, the generator produces the values of S_i from a doubly-truncated normal distribution with $\mu_{[S_i]}$, $\sigma_{[S_i]}$, $l_i = \mu_{[S_i]} - \sigma_{[S_i]}$, and $u_i = \mu_{[S_i]} + \sigma_{[S_i]}$.

Figure 2 shows an example of different datasets with varying degree of heterogeneity. In this example, the number of sub-streams is 10 and the values of *DH* are set to 30%, 50%, and 70%. When *DH* is relatively low (e.g., 30% in Figure 2(a)), we see that most of the sub-streams have *wide* and *similar* spreads of sampling attribute values. The wide spread of each sub-stream indicates that the variability within each sub-stream is high, and the similar spreads among sub-streams indicates that the variability across sub-streams is low. These two combined indicate a low degree of heterogeneity in the entire stream. In



(a) Degree of heterogeneity = 30% (b) Degree of heterogeneity = 50% (c) Degree of heterogeneity = 70%

Fig. 2. Scatter plots of synthetic datasets with different degrees of heterogeneity

contrast, when the *DH* is relatively high (e.g., 70% in Figure 2(c)), we see that most sub-streams have narrow and dissimilar spreads of the sampling attribute values. This is the converse of the Figure 2(a) case above, and thus indicates a high degree of heterogeneity in the entire stream.

Real Datasets. Two kinds of real datasets are used, one (SENS) in the wireless sensor networks application and one (AUCT) in the auction application.

– The SENS real dataset is weather measurements from sensors deployed through the Intel Berkeley Research lab to gather time-stamped topology information, along with humidity, temperature, light and voltage values [1]. SENS is a projection of this data on two attributes, *sensor mote id* and *temperature measurement* acquired from 55 motes. (Data from three motes have incomplete readings and thus have been discarded.). SENS is characterized with a *low* degree of heterogeneity. The low degree of heterogeneity among the temperature readings of different motes is due to the fact that temperatures of nearby regions are expected to be close to each other.
– The AUCT real dataset is for auctions conducted over the Internet through the Federal Communications Commission (FCC) [2]. The entire dataset consists of 55 auction sub-datasets. Each sub-dataset contains bidding information of one auction. We have merged the 55 auction sub-datasets into one single dataset. The order of tuples in the resulting dataset is shuffled and the resulting tuples are projected on two attributes, *auction ID* and *bidding amount.* AUCT is characterized with a *high* degree of heterogeneity. The high degree of heterogeneity of the bidding amounts is intuitive since the bidding amounts can vary to a large extent depending on the auction item.

4.2 Experiment Results

Sample Quality. In this set of experiments, we compare sample accuracy and precision between SRS and CRS. Given that SRS is meant to support both the case of using a sample to obtain the estimate of the entire data stream and the case of using a sample to obtain the estimates of individual sub-streams, experiments are done to report the results in both cases. We refer to the former case as the WHOLE-SAMPLE case and the latter case as the SUB-SAMPLE case[4]. In the SUB-SAMPLE case, the results are reported as the *average square* value of the sample quality metric used. The results of the experiments demonstrate that in both cases SRS outperforms CRS in sample accuracy as well as precision by nearly an order of magnitude.

Whole-sample Case. Figures 3(a) and 3(b) show the SRS accuracy against the CRS accuracy using the synthetic datasets for different degree of heterogeneity and for different number of sub-streams, respectively. Figure 3(a) shows that the degree of heterogeneity has a major influence on the sample accuracy. For a low degree of heterogeneity, (e.g., 10%), we observe that there is only a minor improvement of SRS accuracy over CRS accuracy. The level of improvement, however, increases as the degree of heterogeneity increases. For a high degree of heterogeneity, (e.g., 70% or higher), we see that the SRS accuracy is higher than the CRS accuracy by more than an order of magnitude. The reason for this is that CRS does not consider any heterogeneity between sub-streams whereas SRS does. On the other hand, Figure 3(b) shows that the performance improvement of SRS over CRS is more or less constant regardless of the number of sub-streams. This makes sense because the accuracy of CRS is not affected by the number of sub-streams (Equation 13) and because the accuracy of SRS is primarily influenced by the size and the values of sub-streams (Equation 14).

Figures 4(a) and 4(b) show similar results for the sample precision by demonstrating that the degree of heterogeneity has dominant effect on the precision.

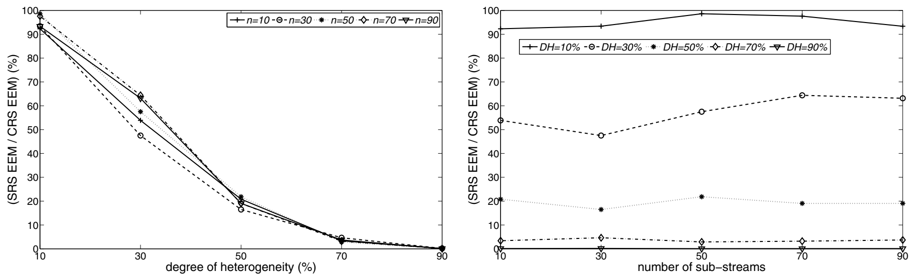
Figure 5 shows the results from using the real datasets AUCT and SENS. The results are consistent with the results from using the synthetic datasets. The figure shows that the improvement of SRS over CRS is higher for for the AUCT dataset than SENS with regard to both sample accuracy and sample precision. This is due to the higher degree of heterogeneity of the AUCT dataset.

Sub-sample Case. Figures 6 and 7 show the results for sub-sample accuracy and sub-sample precision, respectively, using the synthetic dataset. These results report the average square value of EEM (for accuracy) and SE (for precision) per sub-sample. As we see in Figure 6(a), the sub-sample accuracy of SRS improves over the accuracy of CRS linearly with the degree of heterogeneity. Likewise, Figure 7(a) shows a similar trend for the sub-sample precision. From Figure 6(b) and Figure 7(b) we observe that the number of sub-streams is irrelevant to the performance of both SRS and CRS at the level of individual sub-samples.

The results from using the SENS and AUCT real datasets in Figure 8 are similar to those in Figures 6 and 7.

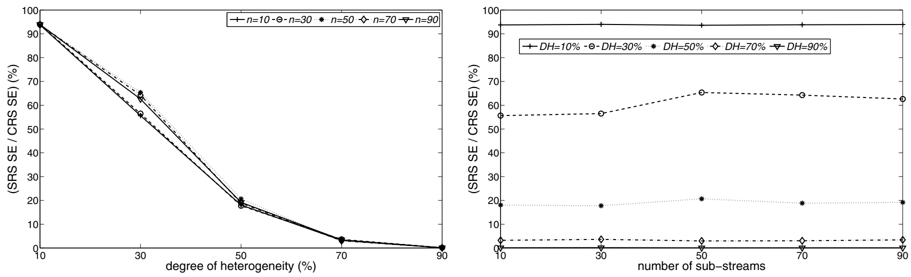
SRS Adaptivity. In this set of experiments, we demonstrate the adaptivity of the SRS by showing the change in the allocation of a stratified reservoir sample as

[4] In the experiments, q is assigned the values of 1 and 0 for the WHOLE-SAMPLE case and the SUB-SAMPLE case, respectively. (Recall Section 3.2).



(a) Effect of the degree of heterogeneity (b) Effect of the number of sub-streams

Fig. 3. WHOLE-SAMPLE accuracy - synthetic datasets



(a) Effect of the degree of heterogeneity (b) Effect of the number of sub-streams

Fig. 4. WHOLE-SAMPLE precision - synthetic datasets

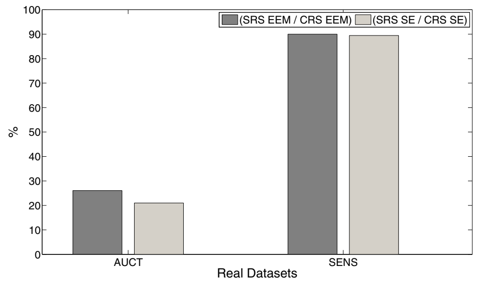
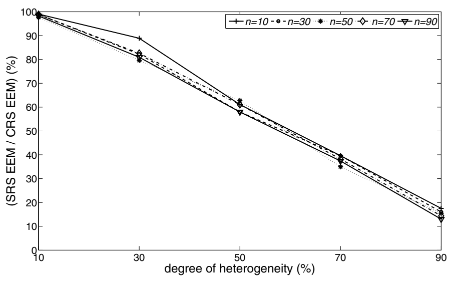
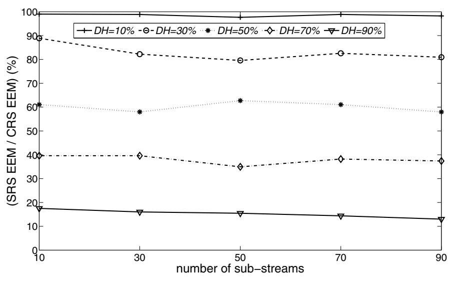


Fig. 5. WHOLE-SAMPLE accuracy and precision - real datasets

a new sub-stream appears in, or an exiting sub-stream expires from, the input stream (i.e., with respect to *data stream membership*) and as the statistical properties of individual sub-streams change over time (i.e., with respect to *sub-streams' stationariness*). Results presented in this section show the change in sub-reservoir sizes over time for five sub-streams synthetically generated and for five sub-streams selected from the AUCT and SENS real datasets. (Only five sub-streams are used for better visibility. Results for a larger number of sub-streams look similar.)

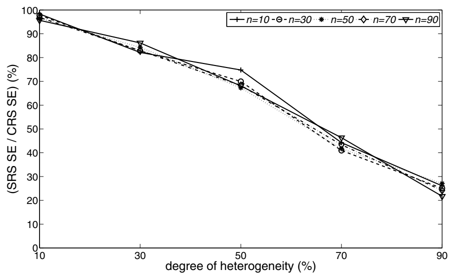


(a) Effect of the degree of heterogeneity

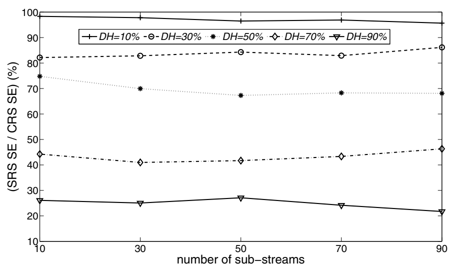


(b) Effect of the number of sub-streams

Fig. 6. SUB-SAMPLE accuracy - synthetic datasets



(a) Effect of the degree of heterogeneity



(b) Effect of the number of sub-streams

Fig. 7. SUB-SAMPLE precision - synthetic datasets

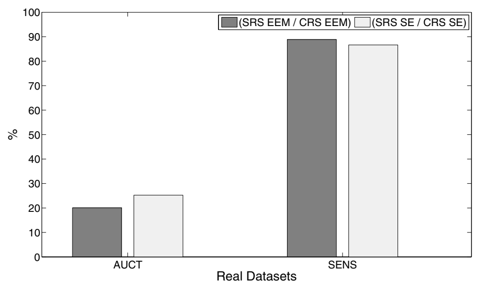
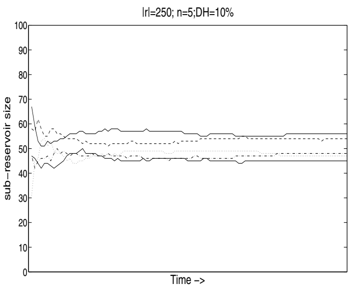


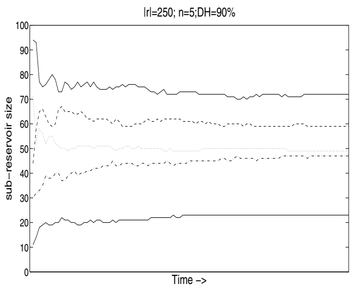
Fig. 8. SUB-SAMPLE accuracy and precision - real datasets

Figure 9 shows the adaptivity of SRS from using synthetic datasets. When *DH* is low (10%) (Figure 9(a)), the sub-reservoir sizes for the sub-streams are relatively close to one another compared with the case of a higher *DH* (90%) (Figure 9(b)). The observed influence of the *DH* on the closeness of the sub-reservoir sizes is reasonable since the allocation of sub-reservoir size is subject to the heterogeneity of the sub-streams.

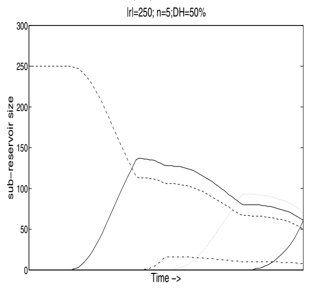
Figures 9(a) and 9(b) also show that the sizes of sub-reservoirs change more frequently in the early stages of sampling and less frequently as the sampling



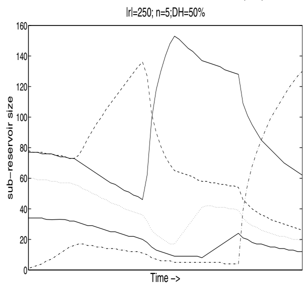
(a) Low degree of heterogeneity



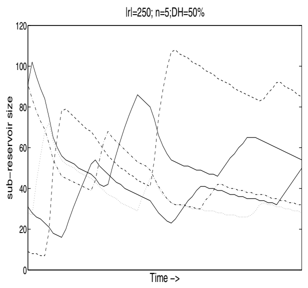
(b) High degree of heterogeneity



(c) Stream membership



(d) Stream non-stationariness: Less frequent change in running statistics



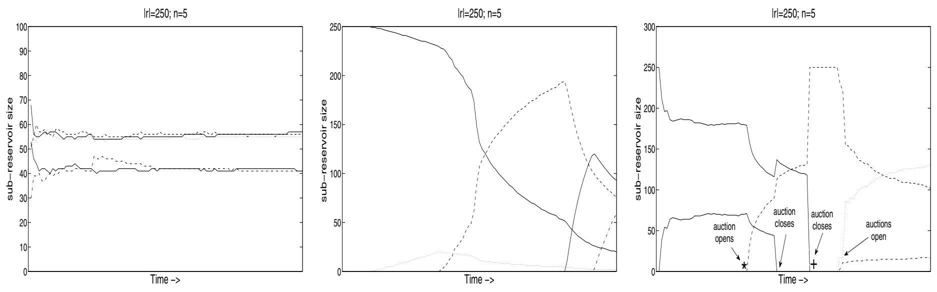
(e) Stream non-stationariness: more frequent change in running statistics

Fig. 9. SRS adaptivity - synthetic datasets

progresses. The frequent change in the early stages is attributed to the significance of the difference in the sub-streams running statistics. As the sampling progresses, the change in a sub-stream statistics relative to the changes in the statistics of other sub-streams becomes smaller and, therefore, does not cause so much frequent changes in sub-reservoir sizes. This trend is in part due to the fact that the underlying sub-streams are stationary in their statistical properties.

In order to conduct experiments to study the influence of data stream membership and non-stationariness, we modify the generation of synthetic datasets as follows. For data stream membership, we make the sub-streams appear in sequence. For non-stationariness, we periodically re-generate n random numbers that correspond to the CVs of n sub-streams such that the overall DH among them is preserved (recall Section 4.1).

Figure 9(c) shows that when a new sub-stream appears in a data stream, the SRS adapts to this situation by releasing memory from the sub-reservoirs of existing sub-streams and allocating the released memory to the sub-reservoirs of the new sub-stream. Figure 9(d) shows that when the running statistics of some sub-streams change over time, SRS decreases (or increases) the sizes of some exiting sub-reservoirs and increases (or decreases) the sizes of other exiting sub-reservoirs. A reduced sub-reservoir size may increase afterwards, and vice versa. The frequency of the change in sub-reservoir sizes is relative to the frequency of the change in the running statistics of sub-streams. (Figure 9(e) shows the case of more frequent change in the running statistics compared to Figure 9(d)).



(a) SENS (b) AUCT (c) AUCT: auctions open and close

Fig. 10. SRS adaptivity - real datasets

Figure 10(a) shows the change of sub-reservoir sizes using SENS real dataset. This dataset represents the case in which sub-streams all exist from the beginning of the input stream and their statistics remain stationary over time. In other words, readings from different sensors scattered to collect temperature information in a certain area are likely to be generated *altogether* from the time the data collection begins. Besides, the change of temperature readings is expected to be similar at any time of the day. Consequently, all sub-reservoir sizes show little change over time.

Figure 10(b) shows the change of sub-reservoir sizes using AUCT real dataset. This dataset represents the case in which sub-streams are added one after another and their statistics change over time. Indeed, in auctions applications, it is unlikely that all auctions (represented by sub-streams) start simultaneously; they are expected to start one after another. Besides, the bidding amount of an auction item naturally increases over time, making the statistics of an auction sub-stream non-stationary. As a consequence, we see significant changes of sub-reservoir sizes over time.

Figure 10(c) further shows the adaptivity of SRS under the scenario of auctions going open and then closed while sampling progresses. When a new auction opens, memory has to be released from existing sub-reservoirs and allocated to the sub-reservoir of the newly opened auction sub-stream (see the point marked with $*$). When an auction closes from further bids (because the auction is forced to close, the auction expires, the auction item is sold, etc.), the sub-reservoir size of the closed auction sub-stream is released and allocated to the sub-reservoirs of the sub-streams of auctions still open (see the point marked with $+$).

5 Related Work

Reservoir sampling technique was proposed by McLeod [15]. Vitter [22] improved the algorithm's performance through more optimization studies. Reservoir sampling has been used in many database applications including clustering [12], data warehousing [7], spatial data management [17], and approximate query processing [23]. Besides the conventional reservoir algorithm, various reservoir-based sampling algorithms have been proposed in the research literature for various applications. Examples of such algorithms include reservoir sampling with replacement (i.e., with duplicates being allowed in the sample) [18], sampling from an evolving dataset (i.e., in the presence of insertions and deletions) [11], biased reservoir sampling (i.e., to bias the sample over time using a given bias function) [4], and adaptive-size reservoir sampling [5] (i.e., to allow the reservoir size to be adjusted in the middle of sampling). In contrast to the existing research on reservoir sampling, our work addresses the problem of *stratifying* a *reservoir sample* rather than maintaining a single reservoir sample.

Stratified sampling has been used for approximate query processing in database systems [3] [8] [9] [13]. *Congressional sampling* [3] proposes to use stratified sampling approach to solve the problem of providing accurate approximate answers of a set of grouped aggregation queries using pre-computed biased samples of the data. In [8], stratified sampling is used in the problem of identifying an appropriate sample selection for answering aggregation queries approximately with the goal of minimizing error in the query result under a given query workload. A comprehensive study of the work proposed in [8] is presented in [9]. The work in [13] solves the problem of using stratified sampling to calculate approximate results of low selectivity aggregation queries. All this work pertains to databases, which makes our work different in addressing *stratified sampling* for *data streams.*

6 Conclusion and Future Work

In this paper, we studied the problem of maintaining a stratified sample over data streams which consist of multiple sub-streams with large statistical variations. First, we discussed the motivation of this new research problem in real-world applications. Second, we discussed an optimal allocation method of a fixed-size reservoir, which can be used whether the sample is needed to generate estimates of the whole data stream or the sub-streams on an individual basis. Third, we presented a sampling algorithm which uses the proposed allocation method to adjust the allocation of a stratified reservoir sample among sub-streams adaptively as sub-streams appear in, or disappear from, the input stream and as their statistical properties change over time. Finally, through experiments, we demonstrated the adaptivity of the proposed algorithm and its superiority over the conventional reservoir sampling algorithm with regard to the sample quality.

Several issues are open for future work. One issue is to extend the proposed algorithm to handle *multi-variate* sampling situation in which an input stream has multiple sampling attributes and an estimate is needed from each sampling attribute. In this situation, it may be required to compromise the allocation of a stratified reservoir sample with respect to the target estimates. Another is to explore the utility of the proposed algorithm in more real-world applications.

Acknowledgments

The authors would like to thank the members of Intel Berkeley Research lab for graciously granting the permission to use their sensor datasets in the experiments. The authors would also like to thank the Federal Communications Commission for making their auctions data available to use in the experiments.

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0415023.

References

1. Intel lab data, `http://berkeley.intel-research.net/labdata/`
2. FCC Auctions, `http://wireless.fcc.gov/auctions/default.htm`
3. Acharya, S., Gibbons, P.B., Poosala, V.: Congressional samples for approximate answering of group-by queries. In: SIGMOD 2000, pp. 487–498 (2000)
4. Aggarwal, C.C.: On biased reservoir sampling in the presence of stream evolution. In: VLDB 2006, pp. 607–618 (2006)
5. Al-Kateb, M., Lee, B.S., Wang, X.S.: Adaptive-size reservoir sampling over data streams. In: SSDBM 2007, pp. 22–33 (2007)
6. Bankier, M.D.: Power allocations: Determining sample sizes for subnational areas. The American Statistician, American Statistical Association 42, 174–177 (1988)
7. Brown, P.G., Haas, P.J.: Techniques for warehousing of sample data. In: ICDE 2006, pp. 6–17 (2006)
8. Chaudhuri, S., Das, G., Narasayya, V.: A robust, optimization-based approach for approximate answering of aggregate queries. In: SIGMOD 2001, pp. 295–306 (2001)
9. Chaudhuri, S., Das, G., Narasayya, V.: Optimized stratified sampling for approximate query processing. ACM Trans. Database Syst. 32(2), 9 (2007)
10. Cochran, W.G.: Sampling Techniques, 3rd edn. John Wiley, Chichester (1977)
11. Gemulla, R., Lehner, W., Hass, P.J.: Maintaining bounded-size sample synopses of evolving datasets. The VLDB Journal 17(2), 173–201 (2008)
12. Guha, S., Rastogi, R., Shim, K.: CURE: an efficient clustering algorithm for large databases. In: SIGMOD 1998, pp. 73–84 (1998)
13. Joshi, S., Jermaine, C.M.: Robust stratified sampling plans for low selectivity queries. In: ICDE, pp. 199–208 (2008)
14. Lohr, S.L. (ed.): Sampling: Design and Analysis. Duxbury Press (1999)
15. McLeod, A., Bellhouse, D.: A convenient algorithm for drawing a simple random sample. Applied Statistics 32, 182–184 (1983)
16. Norgaard, R., Killeen, T.: Expected utility and the truncated normal distribution. Management Science 26, 901–909 (1980)
17. Olken, F., Rotem, D.: Sampling from spatial databases. In: ICDE 2003, pp. 199–208 (2003)
18. Park, B.-H., Ostrouchov, G., Samatova, N.F., Geist, A.: Reservoir-based random sampling with replacement from data stream. In: SDM 2004 (2004)
19. Patel, J.K., Read, C.B.: Handbook of the Normal Distribution. CRC, Boca Raton (1996)
20. Srivastava, U., Widom, J.: Memory-limited execution of windowed stream joins. In: VLDB 2004, pp. 324–335 (2004)
21. LeVeque, R.J., Chan, T.F., Golub, G.H.: Algorithms for computing the sample variance: Analysis and recommendations. The American Statistician, American Statistical Association 37, 242–247 (1983)
22. Vitter, J.S.: Random sampling with a reservoir. ACM Trans. Math. Softw. 11(1), 37–57 (1985)
23. Wu, Y.-L., Agrawal, D., El Abbadi, A.: Query estimation by adaptive sampling. In: ICDE 2002, pp. 639–648 (2002)