ICIC Express Letters
Volume **12**, Number **9**, September **2018**

ICIC International ©2018 ISSN 1881-803X
pp. 915–921

ADAPTIVE STRATIFIED SAMPLING TO SUPPORT HIGH UNIFORMITY CONFIDENCE IN DATA STREAM ENVIRONMENT

Hajin Kim and Yang-Sae Moon*

Department of Computer Science
Kangwon National University
1 Kangwondaehak-gil, Chuncheon-si, Gangwon-do 24341, Korea
hajinkim@kangwon.ac.kr; *Corresponding author: ysmoon@kangwon.ac.kr

Received March 2018; accepted May 2018

Abstract. *In this paper, we focus on improving the accuracy of UC-KSample, which has random sampling characteristics in stream data and at the same time supports high uniformity confidence. To do this, we propose Stratified UC-KSample (S-UCKSample in short) by applying UC-KSample to stratified sampling. In this process, we need to support two features: (1) constant sampling rate and (2) constant uniformity confidence. If we simply apply UC-KSample to stratified sampling, the constant sampling rate is satisfied, but the constant uniformity confidence is not satisfied. We call this an unstable uniformity confidence problem and present the concept of window restart to solve the problem. The window restart makes all substreams create new windows if the uniformity confidence of any substream exceeds the given threshold. Using this window restart, we can always maintain the uniformity confidence above the given threshold. Experimental results show that the proposed S-UCKSample improves the accuracy more than 10 times compared with the basic UC-KSample.*

Keywords: KSample, Uniformity confidence, Stratified sampling, UC-KSample

1. Introduction.

It is very expensive to process a huge volume of continuous stream data in real time [1, 2, 3]. Thus, it is effective to extract and use *samples* that well reflect the characteristics and patterns of the data stream [4, 5]. The sampling technique used in such a streaming environment is divided into (1) *fixed sample size* and (2) *fixed sampling rate* methods according to how to specify the sample amount [6]. In the data stream environment, it is more effective to use the fixed sampling rate at which the amount of samples varies depending on the amount of the input stream. Therefore, in this paper, we focus on improving the accuracy of UC-KSample [7, 8], which has random sampling characteristics among fixed sampling rate methods and supports high uniformity confidence.

In this paper, we apply UC-KSample to *stratified sampling* [9] to improve the accuracy of random sampling in the stream environment. UC-KSample is a random sampling method that maintains the uniformity confidence [10] always over the given threshold while keeping the sampling rate constant in the stream environment. Stratified sampling divides a population into non-duplicated multiple layers and extracts subsamples from each layer so as to provide more accurate samples than single sampling [9]. Thus, we try to apply UC-KSample, which provides high uniformity confidence, to stratified sampling to achieve high sampling accuracy. In the process of applying UC-KSample to stratified sampling, we need to support two features, (1) constant sampling rate and (2) constant uniformity confidence, also in stratified sampling. Based on this observation, in this paper, we propose Stratified UC-KSample (*S-UCKSample* in short) with improved accuracy by extending UC-KSample while satisfying these two features.

DOI: 10.24507/icicel.12.09.915

If we apply UC-KSample to stratified sampling as it is, the constant sampling rate is easily satisfied but the constant uniformity confidence is not satisfied. Applying UC-KSample to stratified sampling, we divide the whole stream into multiple substreams and extract subsamples from UC-KSample in each substream. At this time, if we extract each subsample at a given sampling rate p, we can also maintain the sampling rate of the entire sample, which is the sum of the subsamples, at p or more, and therefore, we can satisfy the feature of constant sampling rate. On the other hand, even though the uniformity confidence of subsamples is higher than the given threshold (ϵ), the uniformity confidence of the entire sample may be lower than or equal to the threshold. In this paper, we call this an *unstable uniformity confidence problem*.

The unstable uniformity confidence problem is due to that there is no criterion for all substreams to create windows at the same time. Thus, in S-UCKSample, we apply the concept of *window restart* to *UC-windows* of UC-KSample [7, 8]. More specifically, for each substream, we use a UC-window to keep its uniformity confidence above the given threshold at all times, but if any substream starts a new window, we restart new windows for all substreams to keep the uniformity confidence of the entire stream above the threshold. In S-UCKSample, if the input stream length of a substream exceeds the window size, all substreams generate new windows and perform sampling, so we can keep not only the uniformity confidence of all the subsamples but also that of the entire sample above the threshold. Experimental results show that the accuracy of the proposed S-UCKSample is 10 times higher than that of the existing UC-KSample.

The rest of the paper is organized as follows. Section 2 describes the related work on uniformity confidence and UC-KSample. Section 3 presents the proposed S-UCKSample and its formal algorithm. Section 4 shows the results of experimental evaluation. Finally, Section 5 concludes the paper.

2. Related Work.

Stratified sampling is a well-known sampling method of dividing a population into non-overlapping multiple layers and then extracting samples from each layer [9]. Since stratified sampling extracts subsamples from each layer to construct the entire sample, it has the advantages of providing higher sample accuracy than random sampling. Figure 1 shows a straightforward application of stratified sampling to the stream environment [11]. As shown in the figure, we first divide an input stream into multiple substreams, then extract samples in each substream, and finally construct an entire sample from those subsamples. Since stratified sampling is more accurate than random sampling, how to efficiently and correctly apply stratified sampling to the stream environment is an important research topic.

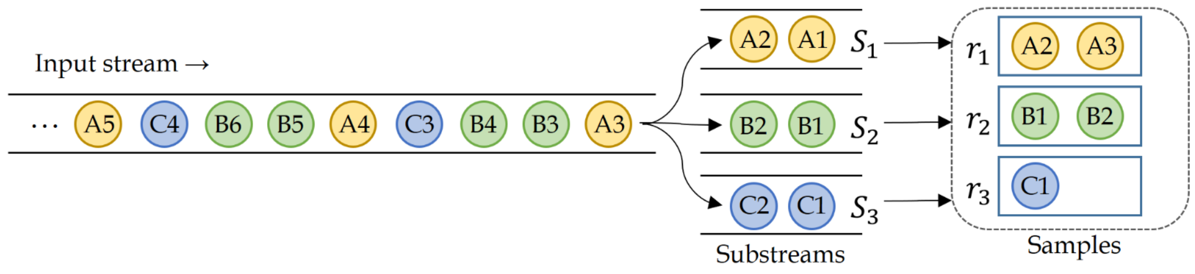

FIGURE 1. Example of applying stratified sampling to the stream environment

UC-KSample [7, 8] is a random sampling method that keeps the sampling rate for the input stream constant and maintains the uniformity confidence [10] above the given threshold. The UC-KSample has two key features: (1) constant sampling rate and (2) constant uniformity confidence. The constant sampling rate is to dynamically increase the sample size so that the sample size remains above $(p \times 100)\%$ of the stream length. The constant uniformity confidence is to ensure that the uniformity confidence of a current sample is always above the user-given threshold (ϵ). Here, the uniformity confidence is an important accuracy measure of a sampling algorithm, which represents how many cases are considered in the sampling process, and it is calculated as Equation (1) [10].

$$UC = \frac{\text{\# of sample cases to be generated by a specific algorithm}}{\text{\# of all possible sample cases to be statistically generated}} \times 100(\%). \qquad (1)$$

However, as the sample size increases dynamically during sampling, the uniformity confidence gradually decreases as the sampling range increases. Thus, in UC-KSample [7, 8], we introduce the concept of UC-windows and use these UC-windows so as to keep the uniformity confidence always above the given threshold. That is, we calculate the window size which keeps the uniformity confidence above the threshold in advance, and we divide the input stream window units to perform sampling. Equation (2) represents this window size calculation method, and please refer to [8] for the detailed description.

$$\epsilon \le \frac{\sum_{x=\max\{0,(\lceil kp\rceil+1)-m\}}^{\lceil kp\rceil} \binom{k}{x}\binom{m}{\lceil kp\rceil+1-x}}{\binom{k+m}{\lceil kp\rceil+1}} \times 100. \qquad (2)$$

Figure 2 shows the algorithm of UC-KSample [8]. The inputs are the sampling ratio p, the input stream *stream*, and the threshold ϵ, and UC-KSample starts the sampling with *reservoir*, an empty sample list. First, we calculate the UC-window size w satisfying the given threshold ϵ by Equation (2) (Line 2). If the length *sLength* of the current stream is 1 or the stream length *wLength* entered in the current window is larger than the window size w, we store the generated samples in the secondary storage, and create a new window to start sampling again (Lines 5 to 7). In Line 8, when a data stream element arrives, we generate a random value *rand* for each element. If the current sample size is smaller than $(p \times wLength)$, we increase the sample size by one to maintain the fixed sampling ratio p, and add the current incoming element to the sample along with the random value (Lines 9 to 11). On the other hand, if the current sample size is not larger than $(p \times wLength)$, we compare the random value of the current element with the element having the smallest random value in the sample list to check whether we replace the smallest one with the

```
Input: p, sampling ratio (p ∈ [0, 1])
       stream, data stream of undefined length
       ε, user-specified threshold of UC-KSample
Output: reservoir[], list of samples
1.  sLength ← 0;    wLength ← 0;
2.  w ← UC-window size by Eq. (2);
3.  while stream != EOF do
4.      sLength++;    wLength++;
5.      if sLength=1 or wLength > w then
6.          reservoir.flush();
7.          wLength ← 1;
8.      rand = Random(0, 1);
9.      if reservoir.size() < (p × wLength) then
10.             reservoir.newSlot();
11.             reservoir.insert(stream[sLength], rand);
12.         else
13.             minimum = reservoir.delete();
14.             if rand > minimum.rand then    reservoir.insert(stream[sLength], rand);
15.             else                           reservoir.insert(minimum);
16.     return reservoir;
```

FIGURE 2. UC-KSample algorithm

current one (Lines 13 to 15). We repeat this process until the stream input ends or the user stops sampling (Lines 3 to 15).

3. S-UCKSample: Stratified UC-KSample.

3.1. Requirement analysis. As shown in Figure 3, S-UCKSample samples each substream with UC-KSample and generates a whole sample by integrating those subsamples. At this time, S-UCKSample should also support two main features, constant sampling rate and constant uniformity confidence, as in UC-KSample. That is, it should maintain the given sampling rate p and uniformity confidence threshold ϵ for an entire sample as well as for each subsample.

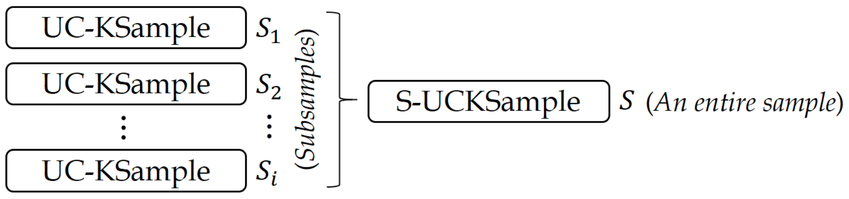

FIGURE 3. UC-KSample algorithm

First, if we apply the sampling rate p to the stratified sampling, the size of each subsample should be $(p \times 100)\%$ of the corresponding substream, and the entire sample size should also be at least $(p \times 100)\%$ of the original entire stream. Since we sample each substream by UC-KSample, the size of the subsample maintains at least $(p \times 100)\%$ of the substream. Here, the total sample is the sum of all subsamples, so its sample size also holds at least $(p \times 100)\%$ of the entire stream. Thus, S-UCKSample maintains the constant sampling rate.

Next, for the constant uniformity confidence, we need to keep the uniformity confidence of an entire sample as well as all subsamples to be above a given threshold (ϵ). In S-UCKSample, we sample each substream by UC-KSample, so we can naturally keep the uniformity confidence of each subsample above the threshold. However, since the data input rate of each substream is different and the window is generated at a different time point for each substream, there is a problem that the uniformity confidence of the entire sample cannot be maintained above the threshold for all the time points. In this paper, we call this an *unstable uniformity confidence problem* and present the concept of *window restart* to solve this problem. To do this, we first calculate the window size that keeps the uniformity confidence of each substream above the given threshold. Then, if the current length of any substream exceeds the window size, all substreams create new windows by the window restart. If any of the substreams has the uniformity confidence below the threshold, all the substreams restart sampling in the *new* window, so the entire sample can also keep the uniformity confidence above the threshold.

3.2. S-UCKSample algorithm. Figure 4 shows the operation procedure of S-UCKSample. Since the entire sample of S-UCKSample consists of several subsamples, we create the windows considering all substreams and generate the slots considering each substream. Therefore, in Figure 4, Steps (3), (4), and (5) proceed to the entire stream, and Steps (6), (7), and (8) proceed to each substream.

Figure 5 shows the proposed algorithm of S-UCKSample. The inputs of S-UCKSample are sampling rate p, stream S, and threshold of uniformity confidence ϵ, and it starts sampling with an empty queue r. First, we calculate the maximum window size w that satisfies the given threshold ϵ by Equation (2) (Line 1). Next, for each data stream element s, we check whether or not the substream S_i belongs to the whole stream S, and if so, we perform the initialization for S (Lines 3-5). Then, we compare the window

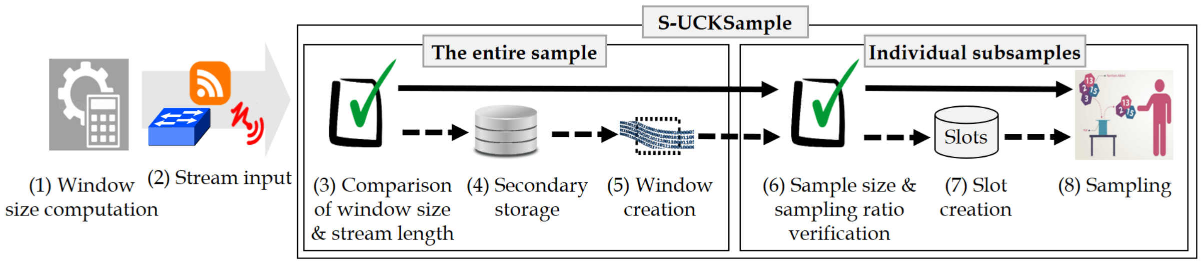

FIGURE 4. Operation procedure of S-UCKSample

```
Input: $p$, sampling ratio ($p \in [0,1]$)
       $S$, data stream of undefined length
       $\varepsilon$, user-specified threshold of S-UCKSample
Output: reservoir $r$, priority queue of samples
1.  $w \leftarrow$ maximum window size that makes UC exceed $\varepsilon$;
2.  for each new tuple $s$ arriving from a substream $S_i$ do
3.      if $S_i \notin S$ then // i.e., $S_i$ is a new substream
4.          $S = S \cup \{S_i\}$;
5.          $wLength_i \leftarrow 0$;
6.      $wLength_i$++;
7.      if $wLength_i > w$ then
8.          for each $S_i \in S$ do
9.              $r_i$.flush();
10.             $wLength_i \leftarrow 0$;
11.     Consider $s$ to include into the subreservoir $r_i$ using UC-Ksample;
12. return $r$;
```

FIGURE 5. S-UCKSample algorithm

size with the data length entered in the corresponding substream (Line 7). If the input data length exceeds the window size, we store the entire sample in the secondary storage, create a new window, and perform sampling by UC-KSample (Lines 7-11). Otherwise, we just proceed sampling to the corresponding substream by UC-KSample (Line 11). We repeat this process until the stream input ends or the user stops sampling (Lines 2-11).

4. Experimental Evaluation.

In this section, we evaluate the accuracy of the proposed S-UCKSample and the existing UC-KSample. Experimental data is nuclear power plant meteorological data of about one million elements, which are collected from five nuclear power plants [12]. Each data element consists of five real number values: temperature, humidity, rainfall, wind speed, and wind direction. Nuclear power plant meteorological data is suitable for stratified sampling experiments because it is composed of multiple substreams of different properties. Since this data is numerical data, we use z-statistics [13] to measure the similarity between the sample and the population. The hardware platform is an HP workstation equipped with Intel i7 3.60GHz CPU and 8.0GB RAM, and the software platform is the Ubuntu 16.04 operating system.

Figure 6 shows the absolute values of z-statistics by changing the uniformity confidence threshold ϵ to 0.650, 0.675, and 0.700, respectively, with a fixed sampling rate $p = 0.01$. In the figure, z-statistic values of both S-UCKSample and UC-KSample are all 0.12 or less, which means that their samples are very similar to the population. However, z-statistic values of S-UCKSample are less than 0.002, which is average 11.5 times lower than those of UC-KSample. It means that the proposed S-UCKSample produces much more accurate samples compared to UC-KSample by using the stratified sampling strategy. In summary

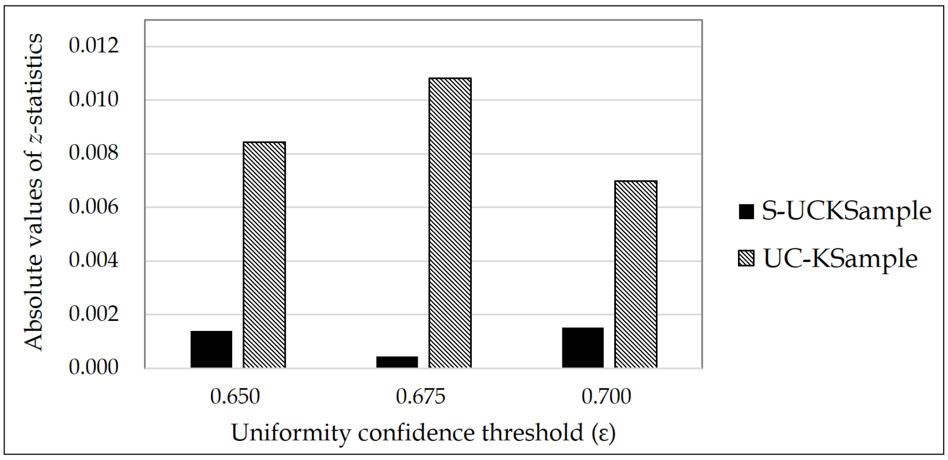

Figure 6. Experimental results of z-statistics with $p = 0.01$ and different ϵ

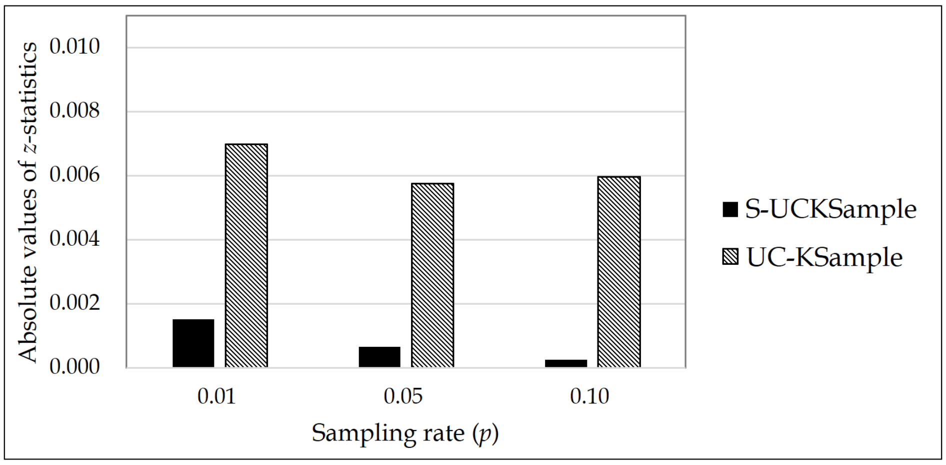

Figure 7. Experimental results of z-statistics with $\epsilon = 0.7$ and different p

of Figure 6, both algorithms produce the samples very similar to the population, but S-UCKSample is much more accurate than UC-KSample.

Figure 7 shows the z-statistics results by changing the sampling rate p to 0.01, 0.05, and 0.10, respectively, with a fixed uniformity confidence threshold $\epsilon = 0.7$. Similar to Figure 6, z-statistic values of both algorithms are all 0.01 or less, so their resulting samples are very similar to the population. Especially, we note that S-UCKSample is much superior to UC-KSample in all cases, and its z-statistics is about 12.2 times lower than that of UC-KSample. As in Figure 6, this is because the proposed S-UCKSample uses the stratified sampling technique while UC-KSample does not. In summary of Figure 7, UC-KSample as well as S-UCKSample provide the samples relatively similar to the population, but S-UCKSample outperforms UC-KSample by an order of magnitude in sampling accuracy.

5. Conclusions.

In this paper, we proposed the *S-UCKSample* (Stratified UC-KSample) algorithm which improved the accuracy of UC-KSample in the stream environment. In order to apply UC-KSample to stratified sampling, we first performed the requirement analysis for applying the constant sampling rate and constant uniformity confidence, which were two key features of UC-KSample, to the stratified sampling. We next derived an *unstable uniformity confidence problem* which occurred when we directly applied the constant uniformity confidence to stratified sampling. To solve this unstable problem, we then presented the concept of *window restart*. Using the window restart we finally proposed S-UCKSample that improved the sampling accuracy of UC-KSample. We empirically showed that the proposed S-UCKSample improved the accuracy more than 10 times compared with the existing UC-KSample. As future work, we focus on improving the uniformity confidence of other sampling algorithms such as hash sampling and priority sampling.

Acknowledgment. This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2016-0-00179, Development of an Intelligent Sampling and Filtering Techniques for Purifying Data Streams).

REFERENCES

[1] B. Babcock, S. Babu, M. Datar, R. Motwani and J. Widom, Models and issues in data stream systems, *Proc. of the 21st ACM Symp. on Principles of Database Systems*, Madison, Wisconsin, pp.1-16, 2002.
[2] J. Leskovec, A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*, Cambridge University Press, 2014.
[3] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak and F. Herrera, A survey on data preprocessing for data stream mining: Current status and future directions, *Neurocomputing*, vol.239, pp.39-57, 2017.
[4] P. J. Haas, Data-stream sampling: Basic techniques and results, *Data Stream Management*, 2016.
[5] C. H. Weiß, Sampling in data mining, *Wiley StatsRef: Statistics Reference Online*, 2014.
[6] G. Cormode and N. Duffield, Sampling for big data: A tutorial, *Proc. of the 20th Int'l Conf. on Knowledge Discovery and Data Mining*, New York, 2014.
[7] H. Kim, S. Lee and Y.-S. Moon, Variable size sampling for maintaining uniformity confidence in the stream environment, *Proc. of Korea Computer Congress 2017*, Jeju, Korea, pp.215-217, 2017 (in Korean).
[8] H. Kim, M.-S. Gil, Y.-S. Moon and M.-J. Choi, Variable size sampling to support high uniformity confidence in sensor data streams, *Int'l Journal of Distributed Sensor Networks*, vol.14, no.4, pp.1-18, 2018.
[9] E. R. Babbie, *The Practice of Social Research*, 14th Edition, Cengage Learning, 2015.
[10] M. Al-Kateb, B. S. Lee and X. S. Wang, Adaptive-size reservoir sampling over data streams, *Proc. of the 19th Int'l Conf. on Scientific and Statistical Database Management*, Banff, Canada, pp.22-33, 2007.
[11] M. Al-Kateb and B. S. Lee, Adaptive stratified reservoir sampling over heterogeneous data streams, *Information Systems*, vol.39, no.1, pp.199-216, 2014.
[12] *Open Data Portal in Korea*, https://www.data.go.kr/main.do?lang=en.
[13] P. S. Mann, *Introductory Statistics*, 9th Edition, Wiley, 2016.