Weighted Random Sampling (2005; Efraimidis, Spirakis)

Pavlos S. Efraimidis, Democritus University of Thrace, utopia.duth.gr/~pefraimi
Paul G. Spirakis, Research Academic Computer Technology Institute, www.cti.gr

entry editor: Paul G. Spirakis

INDEX TERMS: Weighted Random Sampling, Reservoir Sampling, Data Streams, Randomized Algorithms.

1   PROBLEM DEFINITION

The problem of random sampling without replacement (RS) calls for the selection of m distinct random items out of a population of size n. If all items have the same probability to be selected, the problem is known as uniform RS. Uniform random sampling in one pass is discussed in [1, 5, 10]. Reservoir-type uniform sampling algorithms over data streams are discussed in [11]. A parallel uniform random sampling algorithm is given in [9]. In weighted random sampling (WRS) the items are weighted and the probability of each item to be selected is determined by its relative weight. WRS can be defined with the following algorithm D:

Algorithm D, a definition of WRS

Input : A population V of n weighted items
Output :   A set S with a WRS of size m
1 :   For $k = 1$ to m do
2 :   Let $p_i(k) = w_i / \sum_{s_j \in V - S} w_j$ be the probability of item v_i to be selected in round k
3 :   Randomly select an item $v_i \in V - S$ and insert it into S
4 :   End-For

Problem 1 (WRS)**.**
INPUT*: A population V of n weighted items.*
OUTPUT*: A set S with a weighted random sample.*

The most important algorithms for WRS are the Alias Method, Partial Sum Trees and the Acceptance/Rejection method (see [8] for a summary of WRS algorithms). *None of these algorithms is appropriate for one-pass WRS.* In this work, an algorithm for WRS is presented. The algorithm is simple, very flexible, and solves the WRS problem over data streams. Furthermore, the algorithm admits parallel or distributed implementation. To the best knowledge of the entry authors, this is the first algorithm for WRS over data streams and for WRS in parallel or distributed settings.

Definitions: *One-pass WRS* is the problem of generating a weighted random sample in one-pass over a population. If additionally the population size is initially unknown (eg. a data streams), the random sample can be generated with *reservoir sampling* algorithms. These algorithms keep an auxiliary storage, the reservoir, with all items that are candidates for the final sample.

Notation and Assumptions: The item weights are initially unknown, strictly positive reals. The population size is n, the size of the random sample is m and the weight of item v_i is w_i. The function $random(L, H)$ generates a uniform random number in (L, H). X denotes a random variable. Infinite precision arithmetic is assumed. Unless otherwise specified, all sampling problems are without replacement. Depending on the context, WRS is used to denote a weighted random sample or the operation of weighted random sampling.

2 KEY RESULTS

The crux of the WRS approach of this work is given with the following **algorithm A:**
Algorithm A

Input : A population V of n weighted items
Output : A WRS of size m
1: For each $v_i \in V$, $u_i = random(0, 1)$ and $k_i = u_i^{(1/w_i)}$
2: Select the m items with the largest keys k_i as a WRS

Theorem 1. *Algorithm A generates a WRS.*

A reservoir-type adaptation of algorithm A is the following **algorithm A-Res**:
Algorithm A with a Reservoir (A-Res)

Input : A population V of n weighted items
Output : A reservoir R with a WRS of size m
1: The first m items of V are inserted into R
2: For each item $v_i \in R$: Calculate a key $k_i = u_i^{(1/w_i)}$, where $u_i = random\ (0,1)$
3: Repeat Steps 4–7 for $i = m+1, m+2, \ldots, n$
4: $\quad$ The smallest key in R is the current threshold T
5: $\quad$ For item v_i: Calculate a key $k_i = u_i^{(1/w_i)}$, where $u_i = random\ (0,1)$
6: $\quad$ If the key k_i is larger than T, then:
7: $\quad\quad$ The item with the minimum key in R is replaced by item v_i

Algorithm A-Res performs the calculations required by algorithm A and hence by Theorem 1 A-Res generates a WRS. The number of reservoir operations for algorithm A-Res is given by the following Proposition:

Theorem 2. *If A-Res is applied on n weighted items, where the weights $w_i > 0$ are independent random variables with a common continuous distribution, then the expected number of reservoir insertions (without the initial m insertions) is:*

$$\sum_{i=m+1}^{n} P\,[\,item\ i\ is\ inserted\ into\ S\,] \;=\; \sum_{i=m+1}^{n} \frac{m}{i} \;=\; O\!\left(m \cdot \log\!\left(\frac{n}{m}\right)\right)$$

Let S_w be the sum of the weights of the items that will be skipped by A-Res until a new item enters the reservoir. If T_w is the current threshold to enter the reservoir, then S_w is a continuous random variable that follows an exponential distribution. Instead of generating a key for every item, it is possible to generate random jumps that correspond to the sum S_w. Similar techniques have been applied for uniform random sampling (see for example [3]). The following algorithm A-ExpJ is an exponential jumps - type adaptation of algorithm A:
Algorithm A with exponential jumps (A-ExpJ)

Input : A population V of n weighted items
Output : A reservoir R with a WRS of size m
1: The first m items of V are inserted into R

2:   For each item $v_i \in R$: Calculate a key $k_i = u_i^{(1/w_i)}$, where $u_i = $ random (0,1)

3:   The threshold T_w is the minimum key of R

4:   Repeat Steps 5–10 until the population is exhausted

5:   Let $r = random(0,1)$ and $X_w = \log(r)/\log(T_w)$

6:   From the current item v_c skip items until item v_i, such that:

7: $w_c + w_{c+1} + .. + w_{i-1} \ < \ X_w \ \leq \ w_c + w_{c+1} + .. + w_{i-1} + w_i$

8:   The item in R with the minimum key is replaced by item v_i

9:   Let $t_w = T_w{}^{w_i}$, $r_2 = random(t_w, 1)$ and v_i's key: $k_i = r_2{}^{(1/w_i)}$

10:   The new threshold T_w is the new minimum key of R

Theorem 3. *Algorithm A-ExpJ generates a WRS.*

The number of exponential jumps of A-ExpJ is given by Proposition 2. Hence algorithm A-ExpJ reduces the number of random variates that have to be generated from $O(n)$ (for A-Res) to $O(m \log(n/m))$. Since generating high-quality random variates can be a costly operation this is a significant improvement for the complexity of the sampling algorithm.

3   APPLICATIONS

Random sampling is a fundamental problem in computer science with applications in many fields including databases (see [4, 8] and the references therein), data mining, and approximation algorithms and randomized algorithms [6]. Consequently, algorithm A for WRS is a general tool that can find applications in the design of randomized algorithms. For example, algorithm A can be used within approximation algorithms for the k-Median [6].

The reservoir based versions of algorithm A, A-Res and A-ExpJ, have very small requirements for auxiliary storage space (m keys organized as a heap) and during the sampling process their reservoir continuously contains a weighted random sample that is valid for the already processed data. This makes the algorithms applicable to the emerging area of algorithms for processing data streams([2, 7]).

Algorithms A-Res and A-ExpJ can be used for weighted random sampling with replacement from data streams. In particular, it is possible to generate a weighted random sample with replacement of size k with A-Res or A-ExpJ, by running concurrently, in one pass, k instances of A-Res or A-ExpJ respectively. Each algorithm instance must be executed with a trivial reservoir of size 1. At the end, the union of all reservoirs is a WRS with replacement.

4   OPEN PROBLEMS

None is reported.

5   EXPERIMENTAL RESULTS

None is reported.

6   DATA SETS

None is reported.

7 URL to CODE

The algorithms presented in this work are easy to implement. An experimental implementation in Java can be found at: http://utopia.duth.gr/~pefraimi/projects/WRS/index.html

8 CROSS REFERENCES

None is reported. Entry editors please feel free to add some.

9 RECOMMENDED READING

[1] J. H. AHRENS AND U. DIETER, *Sequential random sampling*, ACM Trans. Math. Softw., 11 (1985), pp. 157–169.

[2] B. BABCOCK, S. BABU, M. DATAR, R. MOTWANI, AND J. WIDOM, *Models and issues in data stream systems*, in Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM Press, 2002, pp. 1–16.

[3] L. DEVROYE, *Non-uniform Random Variate Generation*, Springer Verlag, New York, 1986.

[4] C. JERMAINE, A. POL, AND S. ARUMUGAM, *Online maintenance of very large random samples*, in SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, New York, NY, USA, 2004, ACM Press, pp. 299–310.

[5] D. KNUTH, *The Art of Computer Programming*, vol. 2 : Seminumerical Algorithms, Addison-Wesley Publishing Company, second ed., 1981.

[6] J.-H. LIN AND J. VITTER, *ϵ-approximations with minimum packing constraint violation*, in 24th ACM STOC, 1992, pp. 771–782.

[7] S. MUTHUKRISHNAN, *Data streams: Algorithms and applications*, Foundations & Trends in Theoretical Computer Science, 1 (2005).

[8] F. OLKEN, *Random Sampling from Databases*, PhD thesis, Department of Computer Science, University of California at Berkeley, 1993.

[9] V. RAJAN, R. GHOSH, AND P. GUPTA, *An efficient parallel algorithm for random sampling*, Information Processing Letters, 30 (1989), pp. 265–268.

[10] J. VITTER, *Faster methods for random sampling*, Communications of the ACM, 27 (1984), pp. 703–718.

[11] ——, *Random sampling with a reservoir*, ACM Trans. Math. Softw., 11 (1985), pp. 37–57.