Fast Parallel Weighted Random Sampling*

Pavlos S. Efraimidis† Paul G. Spirakis

Computer Technology Institute
Department of Computer Engineering and Informatics
University of Patras
efraimid@cti.gr, spirakis@cti.gr

Abstract

In this work, a new parallel algorithm for drawing a non-uniform random sample of size M from a population of N weighted elements, where $M \leq N$, is presented. The algorithm guarantees that for each element, the relative probability of being selected is determined by its relative weight. The running time of the algorithm is O(logN) on O(N) processors.

*This work was partially supported by ESPRIT LTR ALCOM-IT (contract No. 20244).

†The work of P.S. Efraimidis was supported in part by the Bodosaki Foundation under a grant for PhD studies. Bodosaki Foundation, Leoforos Amalias 20, 10557 Athina, Greece

1 Introduction

The problem of random sampling calls for the selection of M distinct random items out of a population of size N. If all elements of the population have the same probability to be selected, the problem is known as uniform random sampling. A natural generalization of uniform random sampling is the case where the elements are weighted and the relative probability for each item to be selected is proportional to its relative weight. This problem is called weighted random sampling. In this work, we present the first fast parallel algorithm for weighted random sampling.

A natural, and also formal, definition of weighted random sampling is :

Definition 1.1 *Let V be the set of all N items and S be the set containing all the items of the sample. S is initially empty. During the process M items are selected and inserted into S with the following iterative* **process P***:*

1. *Select the 1_{st} item, s_1, from V and insert s_1 to S. For each item $s_i \in V - S$, the probability of being selected as s_1 is:* $p_i(1) = \frac{w_i}{\sum_{s_j \in V} w_j}$
2. *Select the 2_{nd} item, s_2, from V-S, and insert s_2 to S. For each item $s_i \in V - S$, the probability of being selected as s_2 is:* $p_i(2) = \frac{w_i}{\sum_{s_j \in V-S} w_j}$
3. ...
4. *Select the m^{th} item, s_m, from V-S, and insert s_m to S. For each item i of V-S, the probability of being selected as s_m is:* $p_i(m) = \frac{w_i}{\sum_{s_j \in V-S} w_j}$

The set $S = s_1, s_2, ..., s_m$ is a weighted random sample of size M.

Related Work: To the best of our knowledge, no fast parallel algorithm is known for non-uniform random sampling. For the case of uniform random sampling Rajan, Ghosh and Gupta present in [4] a fast parallel algorithm for uniform random sampling. However their algorithm seems not to be appropriate for weighted random sampling. A well studied related problem is that of a non-uniform selection of a single element out of a population of N weighted elements and there are several algorithms known for it. A weighted random sample can be obtained by executing a sequence of M non-uniform selections without replacement. However, the complexity of this approach depends linearly on the size M of the sample and there is no obvious way to parallelize it.

The main result of this work is an efficient parallel algorithm for weighted random sampling of M elements out of a population of size N. The algorithm is simple and runs in log(N) time on O(N) processors.

2 Algorithm A

Algorithm A is the following experiment:

1. Every element i of the population "generates" a random number u_i uniformly distributed in [0,1].
2. Every element calculates its own key $k_i = u_i^{\frac{1}{w_i}}$.
3. The m elements corresponding to the m largest keys form the random sample.

3 Analysis

In this section we will show that Algorithm A is correct and that it runs in logN time on N processors. We will use the following Lemma:

Lemma 3.1 *Let U_1 and U_2 be independent random variables with uniform distributions in [0,1]. Let X_1 and X_2 be the random variables, $X_1 = (U_1)^(1/w_1)$ and $X_2 = (U_2)^(1/w_2)$ were w_1 and w_2 are positive reals. Then $P[X_1 > X_2] = \frac{w_1}{w_1+w_2} = \frac{w_2}{w_1} P[X_1 < X_2]$.*

Proof: $P[X_1 > X_2] = \int_0^1 P[X_1 > \alpha] * P[X_2 = \alpha] * d\alpha = \frac{w_1}{w_1+w_2}$. ■

Every problem instance of weighted random sampling is described by a population of N weighted elements and a positive number M, $0 < M < N$. Algorithm A runs on an instance with N weighted elements and generates a sample of size M.

We will show that Algorithm A is equivalent to the process P. We first we note that, on same problem instance, any possible outcome of the procedure P is also a possible outcome of algorithm A, that is any random sample that can be generated by P can also be generated by A. If additionally every set of M elements occurs with the same probability as an outcome of S and as an outcome of A, then the two processes are equivalent.

We assume that Algorithm A and process P are executed independently on identical problem instances with population V of the same N weighted elements. We let Algorithm A select the random sample S_A and process P generates the weighted random sample S_P. Both sets S_A and S_P are of size M. Then:

Claim 3.1

$$\forall l \, (1 \leq l \leq M) \; : \; P[s_i \, is \, selected \, in \, round \, l \, in \, P] = P[s_i \, has \, the \, l^{th} \, largest \, key \, in \, A]$$

Proof: By induction on l. ■

Claim 3.2 *For every item $s_i \in V$ the probability that s_i is in the sample S_A generated by algorithm A is the same with the probability of s_i being in sample S_P generated by Algorithm P.*

Proof: An item s_i is in S_P if it is selected in one of the M iterations of the process.
Hence $\forall i, P[s_i \in S_P] = \sum_{1 \leq l \leq M} P[s_i \, is \, selected \, in \, round \, l]$.
An item s_i is in S_A if its weighted random key belongs to the M largest keys.
Hence $\forall i, P[s_i \in S_A] = \sum_{1 \leq l \leq M} P[s_i \, has \, the \, l^{th} \, largest \, key]$.
The proof of Claim 3.2 is completed by Claim 3.1. ■

Theorem 3.1 *The outcome of Algorithm A is a weighted random sample of size M.*

Proof: From Claim 3.2. ■

The key generation is done O(1) steps on O(N) processors. The keys are sorted in time O(logN) on O(N) processors. Alternatively a the M_{th} median key is selected in time $O((loglogN)^2)$ on O(N) processors [1].

This proves that:

Theorem 3.2 *Algorithm A runs in time O(logN) on O(N) Processors.*

4 Discussion

In this work, we present a fast parallel algorithm for weighted random sampling of M distinct elements out of a population of N distinct weighted elements. The algorithm for fast parallel weighted random sampling is a general tool that can find applications in the design of any kind of randomized parallel algorithms. We designed and used algorithm A for a fast parallel approximation algorithm for the k-median problem[1] in [2].

References

[1] R. Cole and C.K. Yap. An parallel median algorithm. *Information Processing Letters*, 20:137–139, 1985.

[2] P.S. Efraimidis and P.G. Spirakis. Fast parallel approximations to hard combinatorial optimization problems. Technical Report TR99.06.01, Computer Technology Institute, June 1999.

[3] Juh-Han Lin and J.S. Vitter. ϵ-approximations with minimum packing constraint violation. In *24th ACM Symposium on Theory of Computing*, pages 771–782, 1992.

[4] V. Rajan, R.K. Ghosh, and P. Gupta. An efficient parallel algorithm for random sampling. *Information Processing Letters*, 30:265–268, 1989.

[1] Our algorithm for the k-median problem is based on the filtering and rounding by sampling techniques of [3].