SIAM J. COMPUT.
Vol. 9, No. 1, February 1980

0097-5397/80/0901-0009 $01.00/0

AN EFFICIENT METHOD FOR WEIGHTED SAMPLING WITHOUT REPLACEMENT*

C. K. WONG† AND M. C. EASTON†

Abstract. In this note, an efficient method for weighted sampling of K objects without replacement from a population of n objects is proposed. The method requires $O(K \log n)$ additions and comparisons, and $O(K)$ multiplications and random number generations while the method proposed by Fagin and Price requires $O(K n)$ additions and comparisons, and $O(K)$ divisions and random number generations.

Key words. sampling without replacement, algorithm, computational complexity

1. Introduction. In [1], Fagin and Price consider the following experiment, which can be called weighted sampling without replacement. The experiment is described as the drawing of balls from an urn without replacement. Assume that an urn contains n balls numbered $1, \cdots, n$ and that the probability of drawing ball i is p_i $(i = 1, \cdots, n, \sum_i p_i = 1)$. Suppose that ball i_1 is selected first. Now renormalize the probabilities of the remaining $(n-1)$ balls so that they sum to 1. Thus, the probability of drawing ball j becomes $p_j/(1-p_{i_1})$, for $j \neq i_1$. Select a second ball from the urn, say, ball i_2. Again renormalize the probabilities of the remaining $(n-2)$ balls so that they sum to 1. Thus, the probability of drawing ball j becomes $p_j/(1-p_{i_1}-p_{i_2})$, for $j \neq i_1, i_2$. Continue the process until K balls have been selected. Fagin and Price use this experiment in the Monte Carlo evaluation of a combinatorial sum.

Another application of the experiment occurs in the design of sampling procedures. A sampling system called *multistage sampling with probability proportional to size* (PPS) is discussed in [2, p. 283]. When sampling human, animal, or plant populations, it is often important to first partition the populations into units within which the variation may be expected to be less than it is overall. In two-stage sampling, for example, first a unit is selected and then individuals within that unit are selected at random. In PPS sampling, the selection of the unit is done with the probability of selection proportional to the size of the population of the unit. The selection of K units is to be carried out. After a unit has been selected, it is no longer eligible for later selection. This can be done by the urn experiment considered here if we represent each unit by a ball which has associated probability equal to the fraction of the total population that is contained in the unit.

A special case of the urn experiment occurs when all p_i's are equal and is referred to as "sampling without replacement" in [3, p. 132]. A very efficient method for this case is described in [4, p. 125], which requires $O(K)$ multiplications, additions, random number generations and computations of the floor function $\lfloor\ \rfloor$. The method for the case of unequal p_i's described in [1] requires $O(Kn)$ additions and comparisons and $O(K)$ divisions and random number generations. Another method for the unequal p_i's case, due to D. B. Lahiri, is described in [6, p. 347]. The number of operations required by Lahiri's method is a function of the probabilities $(p_1, \cdots, p_n)$. If the distribution is "flat", performance approaches that of the unweighted algorithm in [4]. On the other hand, if the probabilities, say, follow Zipf's law [7]: $p_i = c/i$ where $c = 1/\sum_{i=1}^{n} i$, then $O(K n/\log n)$ operations are required. In the worst case, $O(K n)$ operations are required. All methods described require $O(n)$ operations of initialization before the drawing of balls can begin.

* Received by the editors July 7, 1978 and in revised form February 5, 1979.
† IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.

In this note, we describe a method for performing weighted sampling that requires, after $O(n)$ initialization operations, $O(K \log n)$ additions and comparisons and $O(K)$ divisions and random number generations. After a sample of K balls has been drawn, the experiment can be restored to its original state by $O(K \log n)$ operations. Thus, this technique is well suited to the Monte Carlo application in [1], where the experiment is repeated many times.

2. Method. The contents of the urn at any time are described by values $\{p'_1, p'_2, \cdots, p'_n\}$. In general, $p'_i = p_i$ if ball i is still in the urn; $p'_i = 0$ if it has been removed. We maintain values $S_i = \sum_{k=1}^{i} p'_k$, $i = 0, \cdots, n$. Let $Q = S_n$ be the sum of the $\{p_i\}$ of the balls remaining in the urn.

The method of drawing a ball is as follows. Choose x with uniform probability from $[0, Q]$. (This step requires generation of a random number and multiplication.) Then find j such that $S_{j-1} \leqq x < S_j$. The ball to be drawn is ball j. It is easy to verify that the probability that ball j is drawn is p'_j/Q.

We now describe an efficient method for implementation. There are two aspects of the implementation that contribute to the efficiency. The first is the method for finding j such that $S_{j-1} \leqq x < S_j$. The second is the method for reflecting the change in value of p'_j to zero in the sums $\{s_j\}$.

3. Initialization. The search for the index of the selected ball is carried out by means of a binary search tree. As part of the initialization procedure, we construct a binary tree having n leaves (external nodes) and having maximum path length from root to leaf of $O(\log n)$. Methods for constructing such trees are described in [5]. The construction requires $O(n)$ operations.

The leaves of the tree are labeled, from left-to-right: $1, 2, \cdots, n$. The other nodes are given arbitrary labels that are distinct from each other and from the leaf labels.

As part of the initialization, we associate values G_i and H_i with internal node i, $i = 1, 2, \cdots, n-1$. Let $L(i)$ be the left-descendent of node i. Let $R(i)$ be the right-descendent of node i.

If node i has a leaf as left-descendent, then define $G_i = p_{L(i)}$. Otherwise, $G_i = G_{L(i)} + H_{L(i)}$. If node i has a leaf as right-descendent, then define $H_i = p_{R(i)}$. Otherwise $H_i = G_{R(i)} + H_{R(i)}$. The computation of the values of $\{G_i\}$ and $\{H_i\}$, carried out from "bottom to top" of the tree, requires $O(n)$ additions.

It is not hard to see that G_i is the sum of values of $\{p_j\}$ that have indices on leaves of the left subtree from node i. (See Fig. 1, where G_i values are shown at each node.) H_i is the sum of values of $\{p_j\}$ that have indices on leaves of the right subtree from node i. (The H_i values are used only in the initialization.)

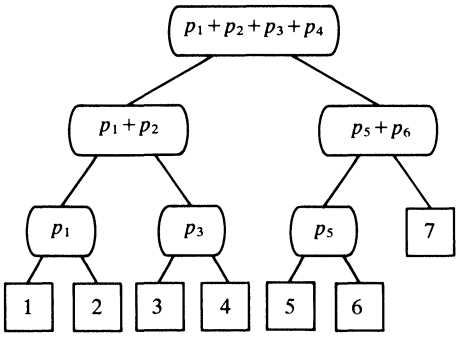

FIG. 1. *Example tree with* $n = 7$ (*sums are* G_i *values*).

At initialization time, G_i also is the sum of values of $\{p'_j\}$ that have indices on leaves of the left subtree from node i. It is this property that will be preserved as balls are drawn.

4. The algorithm. At each step of the algorithm a ball is selected as follows. The first node visited is the root node of the tree. Begin with $C = 0$. At each node i:

if $x < G_i + C$ **then** move to node/leaf $L(i)$.
else set $C = C + G_i$; move to node/leaf $R(i)$.

The procedure ends when a leaf is reached. The label of this leaf gives the index of the chosen ball. $O(\log n)$ additions and comparisons are required for this procedure.

At each node, the value of $C + G_i$ gives the total of $\{p'_i\}$ values in leaves of parts of the tree that lie to the left of the current position.

By the structure of the tree, if the final leaf chosen is leaf j then the value of x satisfies:

$$S_{j-1} = p'_1 + p'_2 + \cdots + p'_{j-1} \leqq x < p'_1 + p'_2 + \cdots + p'_j = S_j.$$

In descending the tree, we keep track of which nodes are departed from *via a left branch.* Once j has been found, the value of G_i for each such node is decremented by the amount p_j in order to reflect the removal of the ball. (This is equivalent to setting the value of p'_j to zero.) The value of Q is also decremented by p_j.

A list is maintained of all changes (index, value) made in $\{G_i\}$. In the course of selecting K balls, at most $O(K \log n)$ entries are made on this list. At the conclusion of the experiment, this list is used to reinitialize the values of $\{G_i\}$. (If K is sufficiently large, of course, it is better to rerun the initialization procedure after each experiment.)

It follows that, once initialization has been completed, additional experiments of drawing K balls from a "full" urn can be performed at the expense of $O(K \log n)$ additions and comparisons and $O(K)$ multiplications and random number generations per experiment.

In summary, the algorithm works as follows to draw the next ball.

(a) Select x uniformly from $[0, Q]$ where Q is the sum of the probabilities of the remaining balls.

(b) Let $C = 0$. Traverse the tree from the root down. At node m, branch right if $x \geqq G_m + C$, left if $x < G_m + C$. On left branches, record the label of the node. On right branches, increment C by G_m.

(c) When leaf j has been reached, decrement by p_j the values of $\{G_i\}$ for the nodes that were listed. The ball drawn is ball j.

Acknowledgment. Our use of sums of probabilities in the tree structure has been simplified by a suggestion of a referee.

REFERENCES

[1] R. FAGIN AND T. G. PRICE, *Efficient calculation of expected miss ratios in the independent reference model*, this Journal, 7 (1978) pp. 288–297.

[2] N. L. JOHNSON AND S. KOTZ, *Urn Models and Their Application*, John Wiley, New York, 1977.

[3] W. FELLER, *An Introduction to Probability Theory and Its Applications*, vol. 1, John Wiley, New York, 1970.

[4] D. KNUTH, *The Art of Computer Programming*, vol. 2, Addison-Wesley, Reading, MA, 1969.

[5] ———, *The Art of Computer Programming*, vol. 3, Addison-Wesley, Reading, MA, 1973.

[6] F. YATES, *Sampling Methods for Censuses and Surveys*, Hafner, New York, 1960.

[7] G. K. ZIPF, *Human Behavior and the Principle of Least Effort*, Addison-Wesley, Reading, MA, 1949.