Statistical Science
1993, Vol. 8, No. 1, 16–19

Approximate Counting via Markov Chains

David Aldous

Abstract. For large finite sets where there is no explicit formula for the size, one can often devise a randomized algorithm that approximately counts the size by simulating Markov chains on the set and on recursively defined subsets.

Key words and phrases: Approximate counting, Markov times, mixing times.

1. EXACT AND APPROXIMATE COUNTING

For a finite set S, there is a close connection between

1. having an explicit formula for the size $|S|$ and
2. having a bounded-time algorithm for generating a uniform random element of S.

As an elementary illustration, we all know that there are $n!$ permutations of n objects. From a proof of this fact, we could write down an explicit 1–1 mapping f between the set of permutations and the set $A = \{(a_1, a_2, \ldots, a_n): 1 \le a_i \le i\}$. Then we could simulate a uniform random permutation by first simulating a uniform random element a of A and then computing $f(a)$. Conversely, given an algorithm that was guaranteed to produce a uniform random permutation after $k(n)$ calls to a random number generator, we could (in principle) analyze the working of the algorithm in order to calculate the chance p of getting the identity permutation. Then we can say that the number of permutations equals $1/p$.

But now suppose we have a hard counting problem for which we cannot find an explicit formula for $|S|$. The purpose of this article is to describe a remarkable technique for constructing randomized algorithms that count S *approximately*. This technique, developed in recent years, has two ingredients. The first idea is that in some settings, having an algorithm for generating an *approximately* uniform random element of S can be used recursively to estimate *approximately* the size $|S|$. We illustrate this with two examples in the next section. The second idea is that we can obtain an approximately uniform random element of S by running a suitable Markov chain on state-space S for sufficiently many steps. This idea and the particular chains used in the two examples are discussed in Section 3. The latter idea is fundamentally similar to the use of Markov chains in the Metropolis algorithm, which underlies the simulated annealing algorithm discussed by Bertsimas and Tsitsiklis in this issue. The difference is that here we seek to simulate a *uniform* distribution rather than a distribution that is deliberately biased toward minima of a cost function. Our Markov chains are simpler to analyze and permit rigorous discussion of polynomial-time convergence issues that seem too hard for rigorous treatment in the context of simulated annealing.

David Aldous is Professor, Department of Statistics, University of California, Berkeley, California 94720.

2. RECURSIVE ESTIMATION OF SIZE

EXAMPLE: VOLUME OF A CONVEX SET (Dyer, Frieze and Kannan, 1989, 1991). Consider the problem of estimating the volume $\mathrm{vol}(K)$ of a convex set K in n-dimensional space, for large n. Suppose we are told that $B(1) \subset K \subset B(r)$, where $B(r)$ is the ball of radius $r = r(n)$. It is believed that there is no polynomial-time deterministic algorithm to approximate $\mathrm{vol}(K)$. But suppose we have a means of simulating, at unit cost, a random point in K whose distribution is approximately uniform. Specify some subset K_1 with $B(1) \subset K_1 \subset K$ and for which $\mathrm{vol}(K_1)/\mathrm{vol}(K)$ is not near 0 or 1. Then by simulating m random points in K and counting the proportion $p(1)$ that fall within K_1,

$$p(1) = \mathrm{vol}(K_1)/\mathrm{vol}(K) + \text{bias} + \text{sampling error}.$$

Now specify a decreasing sequence of such convex subsets

$$K = K_0 \supset K_1 \supset K_2 \cdots \supset K_L = B(1)$$

that will have length $L = O(n \log r)$. Then as above we can estimate each ratio $\mathrm{vol}(K_i)/\mathrm{vol}(K_{i-1})$ as $p(i)$, and finally

$$\text{(1)} \qquad \mathrm{vol}(B(1)) \times \prod_{i=1}^{L} \frac{1}{p(i)} \text{ is an approximation to } \mathrm{vol}(K).$$

How accurate is this approximation? The elementary mathematics of random sampling shows that the sampling error at each step is $O(m^{-1/2})$. So by choosing m to be a large multiple of L^2, the sampling error in (1)

is made small. Note that this makes the total cost $O(L^3)$. The remaining issue is to find a way of simulating an approximately uniform point in K with bias $o(1/L)$, where *bias* is formalized in (2). (See Section 4.) We return to this issue in the next section.

EXAMPLE: MATCHINGS IN A GRAPH (Sinclair and Jerrum, 1989). Fix a graph G with n vertices. In graph-theory terminology, two edges are *independent* if there is no common vertex, and a *matching* is a set of independent edges. Let $M(G)$ be the set of all matchings of G. How can we estimate $|M(G)|$, the number of matchings? Let us show how to do so, given a method of simulating approximately uniform random matchings.

Enumerate the vertices as $s_1, s_2, \ldots, s_n$. Let G_i be the subgraph obtained by deleting $s_1, \ldots, s_i$. If we could simulate a random matching $\mathfrak{M}$ that was exactly uniform (i.e., equally likely to be each of the $|M(G)|$ matchings), then

$$P(s_1 \text{ is in no edge of } \mathfrak{M}) = |M(G_1)|/|M(G)|.$$

So given a method of simulating an approximately uniform matching, we can do m such simulations and record the proportion $p(1)$ of these m simulations in which s_1 is not an edge of the random matching. Then

$$p(1) = |M(G_1)|/|M(G)| + \text{bias} + \text{sampling error}.$$

Similarly, do m simulations of an approximately uniform random matching in G_{i-1}, and use the proportion $p(i)$ in which s_i is not an edge of the matching as an estimator of $|M(G_i)|/|M(G_{i-1})|$. This argument, analogous to (1), leads to

$$\prod_i 1/p(i) \text{ is an approximation to } |M(G)|.$$

These examples share a *self-reducibility* property formalized in Sinclair and Jerrum (1989). Informally, we can relate the size of the set S under consideration to the size of a smaller set S_1 of the same type. The examples illustrate the first idea stated in Section 1. That is, with self-reducible sets, an algorithm for generating an *approximately* uniform random element can be used recursively to estimate *approximately* the size $|S|$.

Three points deserve mention:

1. In the two examples, the self-reducibility property was exact. The reader will rightly suspect that such examples are rare. But the method has been successfully applied where there is only some cruder notion of reducibility, for instance to the problems of approximating the permanent (Jerrum and Sinclair, 1989) and of counting regular graphs (Jerrum and Sinclair, 1988).
2. We should emphasize the point of the product-form estimators: they enable us to estimate exponentially small probabilities in polynomial time. Thus in the first example both $\text{vol}(K)/\text{vol}(B(r))$ and $\text{vol}(B(1))/\text{vol}(K)$ are typically $O(r^{-n})$, so to estimate the ratio in one step would require simulating $O(r^n)$ points instead of $O(n \log r)^3$ points.
3. In principle, we could combine this self-reducibility technique with *any* method of simulating approximately uniform elements of the set, not just the Markov chain method presented in the next section. But in practice no other methods are known for the examples where the technique has been successful.

3. THE MARKOV CHAIN METHOD OF SIMULATING A PROBABILITY DISTRIBUTION

This method is a specialization of the Metropolis algorithm described by Bertsimas and Tsitsiklis in this issue in their article on simulated annealing. Suppose we want to simulate the uniform probability distribution π on a large finite set S. Suppose we can define a Markov chain (picture a randomly moving particle) on states S whose stationary distribution is π. After running the chain for a sufficiently large number of steps, the position of the particle will be approximately uniform. The simplest way to define such a chain is to define a regular graph structure with vertex-set S. In this graph setting, let the Markov chain be a simple random walk on the graph, at each step moving from the current vertex s to a vertex s' chosen uniformly from the neighbors of s.

We now describe graphs for our two examples. To fit the first example into this framework, we discretize in the obvious way, replacing the convex set K by its intersection $S = K \cap \delta Z^n$ with the lattice of edge-length δ, for some small δ. By simulating random walk on S for sufficiently many steps, we will get to a point approximately uniform in S, and hence approximately uniform in K.

In the second example, construct a Markov chain as follows. From a matching $\mathfrak{M}_0$, move to a new matching $\mathfrak{M}_1$ as follows:

1. Pick an edge (v, w) of G at random.
2. If (v, w) is an edge of $\mathfrak{M}_0$, delete it.
3. If v and w are singletons of $\mathfrak{M}_0$, add the edge (v, w).
4. Otherwise, v (say) is a singleton, and w is in an edge (w, u) of $\mathfrak{M}_0$: delete the edge (w, u) and add the edge (v, w).

After sufficiently many steps, the random matching will have approximately uniform distribution.

4. ESTIMATING MIXING TIMES

There is a glaring gap to bridge before we can use the Markov chain method as an honest randomized algorithm. In the simplest use of this method, we will

run the chain for some number k of steps and employ the final state as our "approximately uniform" random element. But how many steps are enough? To justify algorithms, we need bounds on a *mixing time* such as τ_1 or τ_2 below, which formalize the intuitive idea of "number of steps until the distribution is approximately the stationary distribution, independent of starting position." The classic elementary theory of Markov chains says that, under natural conditions (irreducible, aperiodic), the distribution of k steps converges as $k \to \infty$ to the stationary distribution π. But proofs of this elementary general result do not lead to useful bounds in particular examples. The invention of approximate counting, as well as other Markov chain applications such as card-shuffling (Bayer and Diaconis, 1992; Diaconis, 1988) and queueing theory (Blanc, 1988), have motivated recent work of the mathematical issue of estimating mixing times.

In the context of approximate counting, we have some flexibility in the exact choice of the Markov chain, and it is usually possible and very convenient to choose the chain to be *reversible*, that is, to satisfy $\pi(i)P(i, j) = \pi(j)P(j, i)$ for all states i, j. A classical way of formalizing rates of convergence is via the second-largest eigenvalue $\lambda < 1$ of the transition matrix P. This leads to one definition of mixing time as

$$\tau_2 = 1/(1 - \lambda).$$

Another definition of mixing time, more directly applicable in our setting, uses the time taken to make the maximal bias small:

$$\tau_1 = \min\{n : |P_i(X_n \in A) - \pi(A)| \le 1/e \text{ for all } i \in S, A \subset S\}. \tag{2}$$

It is easy to show that after $m\tau_1$ steps the bias is at most $2e^{-m}$, so that a bound on τ_1 enables us to specify confidently a number of steps guaranteed to reduce the bias sufficiently in our applications to approximate counting.

There are several other possible definitions of mixing times, and easy general inequalities giving relations between them. The real issue is not the choice of definition but the development of widely applicable techniques that enable *some* mixing time to be bounded. One interesting technique is to relate mixing times to the quantity

$$c(P) \equiv \max_{A \subset S} \frac{\pi(A)(1 - \pi(A))}{\sum_{i \in A, j \notin A} \pi(i)P(i, j)}.$$

From a bound on $c(P)$, one can use *Cheeger's inequality* (e.g., Sinclair and Jerrum, 1989) to bound τ_2 and then obtain the desired bound on τ_1. Such results are theoretically appealing in that $c(P)$ relates to the "geometry" of the chain, and (for simple random walk on a graph G) to isoperimetric inequalities on G, a quantity of intrinsic graph-theoretic interest. On the other hand, it is usually hard to estimate $c(P)$ well. Diaconis and Stroock (1991) argue that to get upper bounds on τ_2 one can do as well by a direct use of a *distinguished paths* method. For each pair of states, one specifies a path connecting them, seeking to minimize the number of times any fixed edge is used: one can then bound τ_2 in terms of the maximal "probability flow" along any edge. This is still an active research area, and doubtless further techniques will be developed.

We said that the simplest use of this method was to run the chain for some prespecified number of steps and employ the final state as our "approximately uniform" random element. As described in Section 2, this process is then repeated m times (say) in order to calculate the proportion of these m final values that fall in a given subset. Many variations are possible. For instance, instead of the m repetitions one could run the chain for the same total time but estimate the desired probability of a subset by looking at the proportion of *all* times (except an initial segment) that the chain spent in the subset. This is likely by give somewhat of an improvement in practice, though in principle may be no better (Aldous, 1987).

5. CONCLUSION

The topic of approximate counting is still under vigorous development. Let us close with one direction for future research. In addition to the familiar counting problems from the theory of combinatorial algorithms, there is a range of interesting problems arising from physics. We mentioned that the Markov chain method was just a specialization of the Metropolis algorithm for simulating a given probability distribution by inventing a suitable Markov chain. Physicists have long used that algorithm for simulation but typically have been unable to justify rigorously the results of the simulation by proving bounds on the mixing time. For instance, physicists have studied self-avoiding random walks rather intensely by simulation and by nonrigorous mathematical methods. A celebrated open problem is to prove a polynomial-time bound for the mixing time in some algorithm for simulating self-avoiding walks in three dimensions: equivalently, to give a polynomial-time randomized algorithm for approximately counting the number of self-avoiding walks of length n. Bringing together the users of randomized algorithms in physics and the "theoretical" researchers in computer science promises to be a fruitful collaboration.

A much more complete and detailed treatment of the topic of this article can be found in a forthcoming monograph by Alistair Sinclair (1992).

REFERENCES

ALDOUS, D. J. (1987). On the Markov chain simulation method for uniform combinatorial distributions and simulated annealing. *Probability in the Engineering and Informational Sciences* 1 33–46.

BAYER, D. and DIACONIS, P. (1992). Trailing the dovetail shuffle to its lair. *Ann. Appl. Probab.* 2 294–313.

BERTSIMAS, D. and TSITSIKLIS, J. (1993). Simulated annealing. *Statist. Sci.* 8 10–15.

BLANC, J. P. C. (1988). On the relaxation times of open queueing networks. In *Queueing Theory and Its Applications.* CWI Monographs 7, 235–259. North-Holland, New York.

DIACONIS, P. (1988). *Group Representations in Probability and Statistics.* IMS, Hayward, CA.

DIACONIS, P. and STROOCK, D. (1991). Geometric bounds for eigenvalues of Markov chains. *Ann. Appl. Probab.* 1 36–61.

DYER, M., FRIEZE, A. and KANNAN, R. (1989). A random polynomial time algorithm for approximating the volume of convex bodies. In *Proceedings of the 21st ACM Symposium on Theory of Computing* 375–381. ACM Press, New York.

DYER, M., FRIEZE, A. and KANNAN, R. (1991). A random polynomial time algorithm for approximating the volume of convex bodies. *J. Assoc. Comput. Mach.* 38 1–17.

JERRUM, M. and SINCLAIR, A. (1988). Fast uniform generation of regular graphs. *Theoret. Comput. Sci.* 73 91–100.

JERRUM, M. and SINCLAIR, A. (1989). Approximating the permanent. *SIAM J. Comput.* 18 1149–1178.

SINCLAIR, A. J. (1992). *Algorithms for Random Generation and Counting.* Birkhäuser, Boston. To appear.

SINCLAIR, A. and JERRUM, M. (1989). Approximate counting, uniform generation and rapidly mixing Markov chains. *Inform. and Comput.* 82 93–133.