Contemporary Mathematics

Coupling from the Past: a User's Guide

James Propp and David Wilson

Abstract. The Markov chain Monte Carlo method is a general technique for obtaining samples from a probability distribution. In earlier work, we showed that for many applications one can modify the Markov chain Monte Carlo method so as to remove all bias in the output resulting from the biased choice of an initial state for the chain; we have called this method Coupling From The Past (CFTP). Here we describe this method in a fashion that should make our ideas accessible to researchers from diverse areas. Our expository strategy is to avoid proofs and focus on sample applications.

1. Introduction

In Markov chain Monte Carlo studies, one attempts to sample from a distribution π by running a Markov chain whose unique steady-state distribution is π. Ideally, one has proved a theorem that guarantees that the time for which one plans to run the chain is substantially greater than the mixing time of the chain, so that the distribution $\tilde{\pi}$ that one's procedure actually samples from is known to be close to the desired π in variation distance. More often, one merely hopes that this is the case, and the possibility that one's samples are contaminated with substantial initialization bias cannot be ruled out with complete confidence.

The "coupling from the past" procedure introduced in [**PW1**] provides one way of getting around this problem. Where it is applicable, this method delivers samples that are governed by π itself, rather than $\tilde{\pi}$. In the past two years, many researchers have found ways to apply the basic idea in a wide variety of settings. To paraphrase Wilfrid Kendall, one of the first to jump into the fray: "We must now be more ambitious about our simulation objectives. We can no longer be content merely with long-run approximations to equilibrium distributions but instead must strive after perfection" [**K**].

Coupling from the past (hereafter CFTP) is based on what Kendall calls *stochastic flows* (in discrete or continuous time). We restrict attention for now to the discrete-time, finite-state version. Given a finite state-space $\mathbf{X}$, a discrete-time stochastic flow on $\mathbf{X}$ is a random process indexed by discrete time taking

1991 *Mathematics Subject Classification.* Primary 11K45.

The first author was supported by NSA grant MDA904-92-H-3060, NSF grant DMS92-06374, and a career development grant from the M.I.T. Class of 1922.

The second author was supported by an NSF Postdoctoral Fellowship.

its values in the set of maps from $\mathbf{X}$ to itself. If one lets f_i denote the (random) map at time i, then the stochastic flow determines "streamlines" of the form $(x, f_i(x), f_{i+1}(f_i(x)), \ldots)$. Just as the Markov chain Monte Carlo method is predicated on the availability of rapidly-mixing Markov chains, CFTP is predicated on the availability of rapidly-coalescent stochastic flows, in which there is a strong tendency for the streamlines to converge. (Note that once two streamlines have converged, they coincide forever after.)

It is worth stressing at the outset that CFTP is especially valuable as an alternative to standard Markov chain Monte Carlo when one is working with stochastic flows for which one suspects, but has not proved, that rapid coalescence occurs. In such cases, the availability of CFTP makes it less urgent that theoreticians obtain bounds on time-to-coalescence, since CFTP (unlike Markov chain Monte Carlo) cleanly separates the issue of efficiency from the issue of quality of output. That is to say, one's samples are guaranteed to be uncontaminated by initialization bias, regardless of how quickly or slowly they are generated. (See section 7 for a discussion of how to avoid subtle sources of error arising from the variability of the running time of the procedure.)

Even if one has rigorously proved that rapid mixing occurs (as Randall et al. have done [**LRS**] and [**MR**] for the particular application for which CFTP was originally invented), CFTP may be preferable to the usual Markov chain Monte Carlo strategy of running the system until it is "well-mixed" (e.g., running the system until the variation distance between $\tilde{\pi}$ and π is less than 10^{-6}). For one thing, the bounds that have been obtained may be unduly pessimistic. Even if the bounds are tight, CFTP may be the preferred option. Consider, for instance, a situation in which a rigorous upper bound on mixing time has been obtained via coupling methods. On the one hand, the standard Monte Carlo approach would require running the Markov chain for some multiple of the estimated mixing time, and this multiple grows without bound as the desired bound on variation distance shrinks. On the other hand, the fact that the bound on mixing time was proved using coupling techniques is likely to be symptomatic of the existence of a natural way of realizing the Markov chain as part of a stochastic flow whose time-to-coalescence is equal (or comparable) to the mixing time of the chain; if this is the case, one can reduce the initialization bias to zero by running the Markov chain under CFTP for a (random) duration whose expected value is bounded by a fixed multiple of the mixing time.

As an historical aside, we mention that the conceptual ingredients of CFTP were in the air even before the versatility of the method was made clear in [**PW1**]. Precursors include Letac [**Le**], Thorisson [**T**], and Borovkov and Foss [**BF**]. Even back in the 1970's, one can find foreshadowings in the work of Ted Harris (on the contact process, the exclusion model, random stirrings, and coalescing and annihilating random walks), David Griffeath (on additive and cancellative interacting particle systems), and Richard Arratia (on coalescing Brownian motion). One can even see traces of the idea in the work of Loynes [**Lo**] thirty-five years ago.

2. Coupling from the past

Computationally, one needs three things in order to be able to implement the CFTP strategy: a way of generating (and representing) certain maps from $\mathbf{X}$ to itself; a way of composing these maps (so as to be able to simulate the flow for many time-steps); and a way of ascertaining whether total coalescence has occurred, or equivalently, a way of ascertaining whether a certain composite map (obtained by composing many random f's) collapses all of $\mathbf{X}$ to a single element.

The first component is what we call the random map procedure; we model it as an oracle that on successive calls returns independent, identically distributed functions f from $\mathbf{X}$ to $\mathbf{X}$, governed by some selected probability distribution $\mathbf{P}$ (typically supported on a very small subset of the set of all maps from $\mathbf{X}$ to itself). We use the oracle to choose independent, identically distributed maps f_{-1}, f_{-2}, f_{-3}, $\ldots$, f_{-N}, where how far into the past we have to go (N steps) is determined during run-time itself. The defining property that N must have is that the composite map

$$F_{-N}^{0} \stackrel{\text{def}}{=} f_{-1} \circ f_{-2} \circ f_{-3} \circ \cdots \circ f_{-N}$$

must be collapsing. Finding such an N thus requires that we have both a way of composing f's and a way of testing when such a composition is collapsing. (Having the test enables one to find such an N, since one can iteratively test ever-larger values of N, say by successive doubling, until one finds an N that works. Such an N will be a random variable that is measurable with respect to $f_{-N}, f_{-N+1}, \ldots, f_{-1}$.)

Once a suitable N has been found, the algorithm outputs $F_{-N}^{0}(x)$ for any $x \in \mathbf{X}$ (the result will not depend on x, since F_{-N}^{0} is collapsing). We call this output the CFTP sample. It must be stressed that when one is attempting to determine a usable N by guessing successively larger values and testing them in turn, one must use the *same* respective maps f_i during each test. That is, if we have just tried starting the chain from time $-N_1$ and failed to achieve coalescence, then, as we proceed to try starting the chain from time $-N_2 < -N_1$, we must use the same maps f_{-N_1}, f_{-N_1+1}, $\ldots$, f_{-1} as in the preceding attempt. Failure to abide by this principle "voids the warranty" of our algorithm.

We showed in [**PW1**] that, as long as the nature of $\mathbf{P}$ guarantees (almost sure) eventual coalescence, and as long as $\mathbf{P}$ bears a suitable relationship to the distribution π, the CFTP sample will be distributed according to π. Specifically, it is required that $\mathbf{P}$ preserve π in the sense that if a random state x is chosen in accordance with π and a random map f is chosen in accordance with $\mathbf{P}$, then the state $f(x)$ will be distributed in accordance with π.

We will not repeat here the proof, whose simplicity makes it seem rather amazing that the idea was not exploited sooner. Our goal is to show that designing good stochastic flows for CFTP is not much harder than designing good Markov chains for Monte Carlo, and to argue that the Monte Carlo community should be looking for opportunities to apply these ideas. We will give several applications of the CFTP philosophy, and conclude with some words of warning about the proper use of CFTP algorithms.

The article [**PW1**] gives many examples of situations in which CFTP works by virtue of monotonicity. In particular, there is a "top state" $\hat{1}$ and a "bottom state" $\hat{0}$, and the random maps f that have positive probability under $\mathbf{P}$ have the property that a composite map F is collapsing if and only if $F(\hat{0}) = F(\hat{1})$. There are many situations in which this is the case for non-obvious reasons, e.g., the hard-core model on a bipartite graph [**KSW**]. However, in this article we will focus on situations in which this simple state of affairs does not prevail.

For an interesting attempt to embed CFTP in a general probabilistic framework, see the article by Foss and Tweedie [**FT**]. For a discussion of the many uses of the notion of backwards composition of random maps, see the forthcoming survey article by Diaconis [**D**]. For an on-line bibliography on perfect random sampling using Markov chains, see `http://dimacs.rutgers.edu/~dbwilson/exact.html`.

3. The hard-core model

The states of this model are given by subsets of the vertex-set of a finite graph G, or equivalently, by $0,1$-valued functions on the vertex-set. We think of 1 and 0 as respectively denoting the presence or absence of a particle. In a legal state, no two adjacent vertices may both be occupied by particles. The probability of a particular legal state is proportional to λ^m, where m is the number of particles (which depends on the choice of state) and λ is some fixed parameter-value. We denote this probability distribution by π. That is, $\pi(S) = \lambda^{|S|}/Z$ where S is a state, $|S|$ is the number of particles in that state, and $Z = \sum_S \lambda^{|S|}$.

Luby and Vigoda [**LV**] provide a simple Markov chain Monte Carlo procedure for randomizing an initial hard-core state. The random moves they consider are determined by a pair of adjacent vertices u, v and a pair of numbers i, j with (i, j) equal to $(0, 0)$, $(0, 1)$, or $(1, 0)$. They assume that the pair u, v is chosen uniformly from the set of pairs of adjacent vertices in G, and that (i, j) is $(0, 0)$ with probability $\frac{1}{1+2\lambda}$, $(0, 1)$ with probability $\frac{\lambda}{1+2\lambda}$, and $(1, 0)$ with probability $\frac{\lambda}{1+2\lambda}$. Once such a quadruple u, v, i, j is chosen, the algorithm proposes to put a vacancy (resp. particle) at vertex u if i is 0 (resp. 1), and similarly for v and j; if the proposed move would lead to an illegal state, it is rejected, otherwise it is accepted. It is not hard to show that this randomization procedure has π as its unique steady-state distribution.

Luby and Vigoda show that as long as $\lambda \leq \frac{1}{\Delta-3}$, where $\Delta \geq 4$ is the maximum degree of G, this Markov chain is rapidly mixing. They do this by using a coupling argument: two initially distinct states, evolved in tandem, tend to coalesce over time. That is, the authors implicitly embed the Markov chain in a stochastic flow. As such, the method cries out to be turned into a perfect sampling scheme via CFTP.

This is easy to do. One can associate with each *set* of hard-core states a three-valued function on the vertex-set, where the value "1" means that all states in the set are known to have a particle at that vertex, the value "0" means that all states in the set are known to have a vacancy at that vertex, and the value "?" means that it is possible that some of the states in the set have a particle there while others have a vacancy. We can operate directly on this three-valued state-model by means of simple rules that mimic the Luby-Vigoda algorithm on the original two-valued model.

More specifically, we start with a three-valued configuration in which the adjacencies 0–0, 0–?, and ?–? are permitted but in which a 1 can only be adjacent to 0's. Proposals are still of the form $(0, 0)$, $(0, 1)$, $(1, 0)$, and they still have respective probabilities $\frac{1}{2\lambda+1}$, $\frac{\lambda}{2\lambda+1}$, and $\frac{\lambda}{2\lambda+1}$, but proposals are implemented differently. When it is proposed to put 0's at u and v, the proposal is always accepted. When it is proposed to put 0 at u and 1 at v, there are three cases. If all the vertices adjacent to v (other than u) have a 0, the proposal is accepted. If any vertex adjacent to v (other than u) has a 1, the proposal is simply rejected and nothing happens. However, if vertex v has a neighbor (other than u) that has a ? but no neighbor (other than u) that has a 1, then v gets marked with ? and u also gets marked with ? (unless u was already 0, in which case the marking of u does not change). When it is proposed to put 1 at u and 0 at v, the same procedure is followed, but with the roles of u and v reversed.

In short, we can take the work of Luby and Vigoda and, without adding any new ideas, check that their way of coupling two copies of the Luby-Vigoda Markov chain extends to a stochastic flow on the whole state-space. Moreover, this flow can be simulated in such a way that coalescence is easily detected: it is not hard to show that if the 0,1,? Markov chain, starting from the all-?'s state, ever reaches a state in which there are no ?'s, then the Luby-Vigoda chain, using the same random proposals, maps all initial states into the same final state. Hence we might want to call the 0,1,? Markov chain the "certification chain", for it tells us when the stochastic flow of primary interest has achieved coalescence.

One might fear that it would take exponentially long for the certification chain to certify coalescence, but the proof that Luby and Vigoda give carries over straightforwardly to the three-valued setting, and shows that the number of ?'s tends to shrink to zero in polynomial time (relative to the size of the system).

It should be mentioned that Häggström and Nelander [**HN**] have another approach to the hard-core model, in which only one vertex at a time needs to be modified, and that their approach makes use of the very interesting notion of anti-monotone CFTP which has its roots in the work of Kendall on repulsive point-processes. Work of Randall and Tetali [**RT**], in conjunction with the Luby-Vigoda result, implies that the single-site heat-bath Markov chain studied by Häggström and Nelander is also rapidly mixing for $\lambda \leq \frac{1}{\Delta - 3}$.

4. Point-processes with attractive or repulsive interaction

The states of this model are given by discrete subsets of a finite or infinite region of a Euclidean space; for simplicity, we will focus on finite regions. Typically, such processes are defined via a Radon-Nikodym density with respect to the unit-rate Poisson process on the region, and the density rewards or penalizes configurations in which points are close together. (One typical way of measuring clustering is to take the volume of a union of balls of fixed radius, centered on the points in the discrete set; the more the points cluster together, the smaller this volume will be.)

This might seem like an unpromising area for CFTP, since the state space is uncountable. However, there is a kind of monotonicity in the model, and Kendall [**K**] figured out a way to exploit this. (Strictly speaking, the system is monotonic in the attractive case and anti-monotonic in the repulsive case; for now, we focus on the attractive case.)

Kendall effectively constructs a continuous-time stochastic flow. This flow can be modeled as a censored version of a birth/death process whose state at any instant is a point process, and for which the behavior in time is that points get added to the current state, remain alive for an exponentially-distributed random time, and then die. Births are proposed in accordance with Poisson distribution in space-time, but not all proposed births are accepted; proposals are accepted/rejected randomly, with the probability of acceptance being higher when there are one or more "alive" points nearby.

To apply CFTP, one picks an integer N and evolves the system from time $-N$ to time 0, using two different starting states. In one of these states, no particles are present at time $-N$; the other state is generated by an interaction-free birth/death process (call it P) conceived of as running from time $-\infty$ to time 0 with censorship turned off up until time $-N$. The two initial states are evolved in tandem, in accordance with the rules for accepting or rejecting births, using the same random proposals for birth and death events for both cases. If the two simulations agree at time 0, then coalescence occurs over the time-interval $[-N, 0]$ and the state at time 0 can be delivered as the unbiased CFTP sample. If the two simulations disagree, then a larger window $[-N', 0]$ must be tried, using the same realization of the Poisson birth/death process P (this time via its state at time $-N'$) and using the same birth/death proposals on the interval $[-N, 0]$ as before. In practice the realization of the Poisson birth/death process on the window $[-N', 0]$ is built from the one on the window $[-N, 0]$ by simulating the process starting from the state at time $-N$ back into the past until time $-N'$.

Note that in effect one is doing monotone CFTP; the interesting wrinkle is that the "top" state is itself a random variable. It is worth pointing out that in the anti-monotone, repulsive case, a clever variant works, in which one makes a decision about a birth for the *top* simulation in accordance with the environment that the proposed birth faces in the *bottom* simulation, and vice versa. This approach of Kendall was further developed by Häggström and Nelander [**HN**]. Also, Häggström, Van Lieshout, and Møller [**HLM**] have proposed an algorithm for the special case of penetrable spheres that is simpler than Kendall's and works faster, though it is not as amenable to generalization.

It should also be mentioned that CFTP may have theoretical as well as practical significance for such models. For, it gives a way of coupling together different point-processes (by having them hitch a ride on the same Poisson realization). Indeed, what Kendall actually does is associate with each proposed birth a random number chosen uniformly in $[0, 1]$, and the proposal is accepted if and only if the random number falls between 0 and a configuration-specific probability threshold. To get an informative coupling between two models, one should use the same numbers in $[0, 1]$ (called "marks" by Kendall). Such joint realizations of two processes can allow one to prove comparative assertions that might otherwise be quite difficult to establish.

Finally, we mention that the article of Murdoch and Greene [**MG**] has other interesting applications of CFTP to continuous state-spaces.

5. Spanning arborescences with specified root

Let G be a strongly connected directed finite graph with positive weights associated with its arcs. (These assumptions are convenient pedagogically, but they are more stringent than the best theorems require.) A (spanning) arborescence in G with root r is an acyclic directed subgraph of G in which r has out-degree 0 and every other vertex has out-degree 1. We define the weight of an arborescence as the product of the weights of its constituent arcs. This determines a unique probability distribution π on the set of arborescences for which the probability of each arborescence is proportional to its weight. Propp and Wilson [**PW2**] consider the problem of sampling from this distribution π.

A much-studied Markov chain for arborescences can be defined in terms of biased random walk on G, where the transition-probabilities from a vertex v to each of its neighbors are proportional to the weights of the respective arcs connecting v to those vertices. If the current arborescence is rooted at r, one can get a new arborescence by adding an arc from r to s (where s is a random successor of r

under the random walk on G) and deleting the arc that used to emanate from s. It is well known that π is the unique stationary probability measure for this Markov chain on the set of arborescences of G. (This bit of folklore was first published by Anantharam and Tsoucas [**AT**].)

If two arborescences have the same root, there is an obvious way to simulate their future jointly, by having the root take the same random walk in G. This coupling has the property that the two arborescences will continue to have the same root. Moreover, this stochastic flow is rapidly coalescent if the cover-time of the random walk on G is not too large. To see this, note that at any instant, the unique arc emanating from any vertex v other than the current root is the arc that the random walk took on its most recent departure from v. Hence when the random walk has visited all vertices, the two arborescences must have coalesced.

To cash in on this, we define a stochastic flow whose states are arborescences of G rooted at a fixed vertex r. To advance this flow, we perform a random walk on G that starts at r and walks until it next encounters r; we call the walk a random excursion from r to r. While this excursion is going on, we repeatedly update each of the arborescences in the fashion already described, in accordance with the latest step of the random walk. If the excursion happens to cover G, then all the updated arborescences will be identical; this is, the random map associated with this excursion (a map from the set of arborescences rooted at r to itself) is a collapsing map. More generally, if during a succession of excursions the root visits every vertex of G, the corresponding composition of random maps is collapsing.

There is a simple way of describing the effect that an excursion E has on any spanning arborescence A rooted at r. The effect of the excursion can be summarized by an array f indexed by the vertices of the graph. We write $f[v]$ when we think of f as an array and wish to refer to the entry indexed by vertex v, and we write $f(A)$ when we think of f as the "tree-map" associated with the excursion E and wish to refer to the arborescence A' that arises from A as a result of the excursion E. For all vertices v other than r that were visited during excursion E, let $f[v]$ be the vertex of G that was visited immediately following the last visit to v made during the excursion; for all other vertices v, let $f[v]$ be undefined. If $f[v]$ is undefined (and v is not the root), then the excursion never visited v, so the arc out of v in A' is the same as the arc out of v in A. If $f[v] = w$, then the last time the excursion visited v, the next vertex was w, so the arc out of v in A' leads to w. So the array f does indeed contain all the information about the tree-map defined by the excursion from the root to itself. The tree-map defined by any finite sequence of excursions is similarly representable by an array indexed by the vertices of the graph.

To obtain a procedure for randomly generating spanning arborescences, one needs not only a way to represent and randomly generate tree-maps, but also a way to compose them, and a way to detect when a tree-map is a collapsing map. Composing tree-maps is easy: The combined effect of two excursions with respective tree-maps f and g (with the g-excursion preceding the f-excursion in time) is the tree-map $f \circ g$, where the array entry $(f \circ g)[v]$ is equal to $f[v]$ unless $f[v]$ was undefined (corresponding to the situation in which the excursion associated with f did not visit v), in which case $(f \circ g)[v]$ is equal to $g[v]$ (which may itself be undefined). Finally, detecting when coalescence has occurred is simple: one merely checks that $f[v]$ is defined for all v other than the root. Hence we can use CFTP to generate random rooted spanning trees, via prepended excursions; further details can be found in [**PW2**].

If one wants to use CFTP to generate a random spanning arborescence of a graph *without* specifying a root, then one might think one needs a way to couple histories in which two arborescences start out with different roots. That is, one might try to do away with the notion of excursions and simply let the stochastic flow advance one step at a time, so that eventually the roots coincide. Such an approach might be workable, but we have proposed a different way. Specifically, in [**PW2**] we suggest that one first choose a random vertex r to serve as the root, where r is governed by the steady-state distribution for the random walk on G, and then apply the procedure for finding a random arborescence with root r. A CFTP-based procedure for choosing such an r is described in the next section (although the non-CFTP-based method called cycle-popping, described in [**PW2**], is likely to be superior in most applications).

We also mention that in the case where one knows the transition probabilities for the time-reversal of the random walk on G, the procedure for determining the latest starting time $-N$ that leads to coalescence by time 0 is simple: one simply runs the walk backwards from time 0 into the past until all vertices have been visited. Two such cases are (unweighted) random walk on an undirected graph and (unweighted) random walk on an Eulerian directed graph (a directed graph in which each vertex has its in-degree equal to its out-degree). In the first case, the CFTP algorithm is effectively equivalent to the random walk method of Aldous [**A1**] and Broder [**B**]; in the second case, the CFTP algorithm is effectively equivalent to the random walk method of Kandel, Matias, Unger, and Winkler [**KMUW**].

Finally, we stress that CFTP is only one of many schemes for generating random spanning arborescences; see [**PW2**] for others.

6. Random state of an unknown Markov chain

Now we come to a problem that in a sense encompasses all the cases we have discussed so far: the problem of sampling from the steady-state distribution $\pi(\cdot)$ of a general Markov chain. Of course, in the absence of further strictures this problem admits a trivial "solution": just solve for the steady-state distribution analytically! In the case of the systems studied in sections 3 through 5, this is not practical, since the state spaces are large. We now consider what happens if the state space is small but the analytic method of simulation is barred by imposing the constraint that the transition probabilities of the Markov chain are unknown: one merely has access to a black box that simulates the transitions.

It might seem that, under this stipulation, no solution to the problem is possible, but in fact a solution was found by Asmussen, Glynn, and Thorisson [**AGT**]. However, their algorithm was not very efficient. Subsequently Aldous [**A2**] and Lovász and Winkler [**LW**] found faster procedures (although the algorithm of Aldous involves controlled but non-zero error). The CFTP-based solution given below is even faster than that of Lovász and Winkler.

For pictorial concreteness, we envision the Markov chain as biased random walk on some directed graph G whose arcs are labeled with weights, where the transition probabilities from a given vertex are proportional to the weights of the associated arcs (as in the preceding section). We denote the vertex set of G by $\mathbf{X}$, and denote the steady-state distribution on $\mathbf{X}$ by π. Propp and Wilson [**PW2**] give a CFTP-based algorithm that lets one sample from this distribution π.

Our goal is to define suitable random maps from $\mathbf{X}$ to $\mathbf{X}$ in which many states are mapped into a single state. We might therefore define a random map from $\mathbf{X}$ to itself by starting at some fixed vertex r, walking randomly for some large number N of steps, and mapping all states in $\mathbf{X}$ to the particular state v that one has landed in after N steps. However, v is subject to initialization bias, so this random map procedure typically does not preserve π in the sense defined in section 2.

What actually works is a multi-phase scheme of the following sort: Start at some vertex r and take a random walk for a *random* amount of time T_1, ending at some state v; then map every state that has been visited during that walk to v. In the second phase, continue walking from v for a further random amount of time T_2, ending at some new state v'; then map every state that was visited during the second phase but not the first to v'. In the third phase, walk from v' for a random time to a new state v'', and map every hitherto-unvisited state that was visited during that phase to the state v''. And so on. Eventually, every state gets visited, and every state gets mapped to some state. Such maps, like tree-maps, are easy to compose, and it is easy to recognize when such a composition is coalescent (it maps every state to one particular state).

There are two constraints that our random durations T_1, T_2, ... must satisfy if we are planning to use this scheme for CFTP. (For convenience we will assume henceforth that they T_i's are i.i.d.) First, the distribution of each T_i should have the property that, at any point during the walk, the (conditional) expected time until the walk terminates does not depend on where one is or how one got there. This ensures that the stochastic flow determined by these random maps preserves π. Second, the time for the walk should be neither so short that only a few states get visited by the time the walk ends nor so long that generating even a single random map takes more time than an experimenter is willing to wait. Ideally, the expected duration of the walk should be on the order of the cover-time for the random walk. Propp and Wilson [**PW2**] show that by using the random walk itself to estimate its own cover-time, one gets an algorithm that generates a random state distributed according to π in expected time at most 15 times the cover time.

At the beginning of this section, we said that one has access to a black box that simulates the transitions. This is, strictly speaking, ambiguous: Does the black box have an "input port" so that we can ask it for a random transition from a specified state? Or are we merely passively observing a Markov chain in which we have no power to intervene? This ambiguity gives rise to two different versions of the problem, of separate interest. Our CFTP algorithm works for both of them.

For the "passive" version of the problem, it is not hard to show that no scheme can work in expected time less than the expected cover time of the walk, so in this setting our algorithm runs in time that is within a constant factor of optimal. It is possible to do better in the active setting, but no good lower bounds are currently known for this case.

7. Caveats

Of course a major warning that should come on the "packaging" of all algorithms that purport to generate random objects is that we do not know of any source of truly random bits, and that if we found one we would not be able to prove that it was random. But leaving that deep issue aside, there are other matters that need to be addressed.

Any time one uses a sampling scheme in which the running time is random, one runs the risk of introducing bias if one is not careful. CFTP is no different in this regard than most other schemes for randomly generating combinatorial or geometric objects. Clearly if the current run of the algorithm has been running for 1000 CPU hours and shows no sign of terminating, the experimenter would be likely to abort the run; and, in effectively discarding the output that the procedure *would* have generated, had it been allowed to run through to completion, the experimenter would risk biasing his set of samples from $\mathbf{X}$ in favor of those for which the procedure tends to finish more quickly.

One way of dealing with this problem would be to devise algorithms for which the (random) running-time is independent (or nearly independent) of the running time; interrupting such an algorithm and discarding the current output would then contaminate one's collection of samples with no (or hardly any) bias. For a significant step in this direction, see the article of Fill [**F**].

Another way to address the problem of bias caused by user impatience is given by work of Glynn and Heidelberger [**GH**]. Under their approach, an experimenter generates as many samples $x_1, x_2, \ldots, x_n$ as she can during some pre-selected time-interval, subject to the condition that if no samples at all were generated during that interval, she must continue her simulation until one sample is generated (giving $n = 1$). This approach avoids bias in the sense that, for any function f on the space from which one is sampling, the sample-average $(f(x_1) + f(x_2) + \cdots + f(x_n))/n$ is an unbiased estimate for the mean value of $f(x)$.

We suggest that the thing to do when using algorithms in which the correlation between output and running time is unknown is to handle the premature termination problem preventatively: one should perform preliminary assays that enable one to design an experiment that is likely to finish tolerably soon. The probability that the duration of the experiment will exceed one's estimated running-time by a factor of k falls off rapidly in k. Thus the experimenter can confidently commit herself to running the experiment through to completion. Of course, there is always a chance that the experiment will take so long to finish that the experimenter, despite her intentions, will go back on her word and abort (and re-design) the experiment. However, this probability can be made as minuscule as the experimenter desires if she takes the time to pre-test the design of the experiment.

Even in the case where a run is interrupted, through deliberate user impatience or technical mishap, all is not lost. For instance, suppose one is using perfect simulation as a way of estimating the mean value of some quantity $f(x)$ via the sample-average $(f(x_1) + f(x_2) + \cdots + f(x_n))/n$. If one is willing to replace the sample-average by an interval estimate, one can replace the unknown value $f(x_n)$ (corresponding to the run that was terminated prematurely) by the interval whose endpoints are upper and lower bounds on this quantity; in many cases, good two-sided bounds on $f(x_n)$ may be available from the terminated run. We suggest that a user of coupling from the past take the point of view that a prematurely terminated run is not a failed run but a run that determined partial information about a particular random object, namely, the random object that she would have obtained if the run had gone through to completion. Indeed, coupling from the past can be envisioned as a process in which more and more information about a random object is obtained until one finally has determined it completely; it is natural to extend this to situations in which one is free to decide, adaptively, that one's partial information is sufficient for the purpose at hand.

References

[A1] D. Aldous, *A random walk construction of uniform spanning trees and uniform labelled trees*, SIAM Journal on Discrete Mathematics **3** (1990), 450–465.

[A2] D. Aldous, *On simulating a Markov chain stationary distribution when transition probabilities are unknown*, in: D. Aldous, P. Diaconis, J. Spencer, and J. M. Steele, editors, "Discrete Probability and Algorithms", volume 72 of IMA Volumes in Mathematics and its Applications, 1–9, Springer-Verlag 1995.

[AGT] S. Asmussen, P. Glynn, and H. Thorisson, *Stationary detection in the initial transient problem*, ACM Transactions on Modeling and Computer Simulation **2** (1992), 130–157.

[AT] V. Anantharam and P. Tsoucas, A proof of the Markov chain tree theorem, *Statistics and Probability Letters* **8** (1989), 189–192.

[BF] A. A. Borovkov and S. G. Foss, *Stochastically recursive sequences and their generalizations*, Siberian Advances in Mathematics **2** (1992), 16–81.

[B] A. Broder, *Generating random spanning trees*, in: "30th Annual Symposium on Foundations of Computer Science" (1989), 442–447.

[D] P. Diaconis, *Backwards composition of random maps*, survey article in preparation.

[F] J. A. Fill, *An interruptible algorithm for perfect sampling via Markov chains*, preprint (1997); to appear in Annals of Applied Probability.

[FT] S. G. Foss and R. L. Tweedie, *Perfect simulation and backward coupling*, to appear in Stochastic Models.

[GH] P. W. Glynn and P. Heidelberger, *Bias properties of budget constrained simulations*, Operations Research **38** (1990), 801–814.

[HLM] O. Häggström, M. N. M. Van Lieshout, and J. Møller, *Characterisation results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes*, Technical Report R-96-2040, Aalborg University (1996).

[HN] O. Häggström and K. Nelander, *Exact sampling from anti-monotone systems*, preprint (1997).

[KMUW] D. Kandel, Y. Matias, R. Unger, and P. Winkler, *Shuffling biological sequences*, Discrete Applied Mathematics **71** (1996), 171–185.

[K] W. S. Kendall, *Perfect simulation for the area-interaction point process*, to appear in Probability Perspective.

[KSW] J. H. Kim, P. Shor, and P. Winkler, *Random independent sets*, article in preparation.

[Le] G. Letac, *A contraction principle for certain Markov chains and its applications*, Contemporary Mathematics **50** (1986), 263–273.

[LW] L. Lovász and P. Winkler, *Exact mixing in an unknown Markov chain*, Electronic Journal of Combinatorics **2** (1995), paper #R15.

[Lo] R. M. Loynes, *The stability of a queue with non-independent inter-arrival and service times*, Proceedings of the Cambridge Philosophical Society **58** (1962), 497–520.

[LRS] M. Luby, D. Randall, and A. Sinclair, *Markov chain algorithms for planar lattice structures*, in: Proceedings of the 36th IEEE Symposium on Foundations of Computing (1995), 150–159.

[LV] M. Luby and E. Vigoda, *Approximately counting up to four* (extended abstract), in: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing (1995), 150–159.

[MR] N. Madras and D. Randall, *Factoring graphs to bound mixing rates*, in: Proceedings of the 37th IEEE Symposium on Foundations of Computing (1996), 194–203.

[MG] D. Murdoch and P. Green, *Exact sampling from a continuous state space*, to appear in the Scandinavian Journal of Statistics.

[PW1] J. Propp and D. Wilson, *Exact sampling with coupled Markov chains and applications to statistical mechanics*, Random Structure and Algorithms **9** (1996), 223–252.

[PW2] J. Propp and D. Wilson, *How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph*, to appear in the Journal of Algorithms (SODA '96 special issue).

[RT] D. Randall and P. Tetali, *Analyzing Glauber dynamics by comparison of Markov chains* (extended abstract), preprint (1997).

[T] H. Thorisson, *Backward limits*, Annals of Probability **16** (1988), 914–924.

Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

E-mail address: propp@math.mit.edu

Institute for Advanced Study, Princeton, New Jersey 08540

E-mail address: dbwilson@math.ias.edu