The Annals of Applied Probability
1998, Vol. 8, No. 1, 131–162

AN INTERRUPTIBLE ALGORITHM FOR PERFECT SAMPLING VIA MARKOV CHAINS[1]

BY JAMES ALLEN FILL

Johns Hopkins University

For a large class of examples arising in statistical physics known as *attractive spin systems* (e.g., the Ising model), one seeks to sample from a probability distribution π on an enormously large state space, but elementary sampling is ruled out by the infeasibility of calculating an appropriate normalizing constant. The same difficulty arises in computer science problems where one seeks to sample randomly from a large finite distributive lattice whose precise size cannot be ascertained in any reasonable amount of time.

The Markov chain Monte Carlo (MCMC) approximate sampling approach to such a problem is to construct and run "for a long time" a Markov chain with long-run distribution π. But determining how long is long enough to get a good approximation can be both analytically and empirically difficult.

Recently, Propp and Wilson have devised an ingenious and efficient algorithm to use the same Markov chains to produce *perfect* (i.e., exact) samples from π. However, the running time of their algorithm is an unbounded random variable whose order of magnitude is typically unknown a priori and which is not independent of the state sampled, so a naive user with limited patience who aborts a long run of the algorithm will introduce bias.

We present a new algorithm which (1) again uses the same Markov chains to produce perfect samples from π, but is based on a different idea (namely, acceptance/rejection sampling); and (2) eliminates user-impatience bias. Like the Propp–Wilson algorithm, the new algorithm applies to a general class of suitably monotone chains, and also (with modification) to "anti-monotone" chains. When the chain is reversible, naive implementation of the algorithm uses fewer transitions but more space than Propp–Wilson. When fine-tuned and applied with the aid of a typical pseudorandom number generator to an attractive spin system on n sites using a random site updating Gibbs sampler whose mixing time τ is polynomial in n, the algorithm runs in time of the same order (bound) as Propp–Wilson [expectation $O(\tau \log n)$] and uses only logarithmically more space [expectation $O(n \log n)$, vs. $O(n)$ for Propp–Wilson].

1. Overview.

In this paper we will present an efficient new algorithm for perfect (i.e., exact) distributional sampling that combines the ideas of rejection sampling and Markov chain Monte Carlo (MCMC). In Section 2 we describe a large class of examples (namely, attractive spin systems) arising in statistical physics and computer science where elementary exact sampling from a distri-

Received December 1996; revised May 1997.

[1] Supported in part by NSF Grants DMS-93-11367 and DMS-96-26756.

AMS 1991 *subject classifications*. Primary 60J10; secondary 68U20, 60G40, 62D05.

Key words and phrases. Markov chain Monte Carlo, perfect simulation, rejection sampling, monotone chain, attractive spin system, Ising model, Gibbs sampler, separation, strong stationary time, duality, partially ordered set.

bution of interest is desirable but infeasible due to the virtual incomputability of a normalizing constant. The usefulness of MCMC for such problems is explained in Section 3, and background material on monotonicity is discussed in Section 4. In Section 5 we review a stationary sampling scheme for monotone chains recently developed by Propp and Wilson that is based on backward coupling, and we point out a subtle bias introduced by naive use of their algorithm. In Section 6 we assess the extent of this bias and analyze the scheme's time and space requirements. In Section 7 we present, and in Section 8 we analyze, a new algorithm based on rejection sampling which eliminates the bias. As discussed in Section 9, this algorithm is closely connected with the construction of a certain minimal strong stationary time.

A suitable modification of either algorithm allows for the treatment of the "anti-monotone" chains described in [22]; see [26] for a pioneering application of the Propp–Wilson algorithm in this direction. (In passing, and in connection with [26], we note that the spatial statistics community has fallen on the idea of perfect simulation with avid enthusiasm; see also [23] and [27].)

For reversible chains, the new algorithm is faster, in the precise sense that a bound on the expected number of transitions is of smaller order of magnitude than the corresponding bound for the backward coupling algorithm. However, the new algorithm's memory space requirement turns out to be too large for practical application to attractive spin system problems; in Section 10 we fine-tune the algorithm for these problems to make it competitive in time and space requirements with the backward coupling algorithm.

In Section 11 we give a detailed comparison of the Propp–Wilson algorithm and our new one. There one will in particular find corroboration of the assertions in the final two sentences of the abstract. A reader familiar with the Propp–Wilson algorithm and its applications and uninterested in proofs or theoretical details might wish to proceed directly to Sections 7.2, 10.2 and the summary 11.2.

Notation. Throughout this paper, we write $\mathscr{L}(\mathbf{Z})$ for the probability distribution (or *law*) of a random variable $\mathbf{Z}$ [or, more generally, measurable mapping $\mathbf{Z}$ from a probability space $(\Omega, \mathscr{A}, P)$ into any measurable space $(\Omega', \mathscr{A}')$]. A standard measure of discrepancy between two such laws is the *total variation* (or just *variation*) distance, a worst-case absolute error:

$$\|\mathscr{L}(\mathbf{Z}) - \mathscr{L}(\mathbf{Z}')\| := \sup_{\mathbf{A}\in\mathscr{A}'} |P(\mathbf{Z}\in\mathbf{A}) - P(\mathbf{Z}'\in\mathbf{A})|.$$

We denote the complement of an event B by B^c. Let a_n, $b_n > 0$; as usual, $a_n = O(b_n)$ means that $\sup_n(a_n/b_n) < \infty$; $a_n = \Omega(b_n)$ means that $b_n = O(a_n)$; and $a_n = \Theta(b_n)$ means that $a_n = O(b_n)$ and $b_n = O(a_n)$. We write lg for binary logarithm, ln for natural logarithm, and log when the base does not matter [as in $a_n = O(\log n)$].

2. A case for MCMC: attractive spin systems.

The class of examples known as *attractive spin systems* (cf. [28]) nicely illustrates the usefulness of finite-state MCMC methods. This is a class of statistical physics models which includes the *Ising model* of ferromagnetism and is easily extended to include the more general *Potts model*. See [31], whose terminology we follow, for a superbly written (and more detailed) discussion of these models and their central importance in the study of statistical mechanics, with pointers to the vast literature. The same models are also used in image processing for noise reduction [8, 5, 21, 31]. Other frequently studied attractive spin systems are the *voter model* (e.g., [3], Chapter 14) and *contact process* (e.g., [28]).

Before proceeding to definitions, we note a connection with problems in computer science. Propp and Wilson [31] have shown how the uniform distribution on the finite distributive lattice of order ideals (equivalently, of antichains) of a given partially ordered set can be viewed as an attractive measure π on a certain spin system. Thus methods we shall describe in Sections 5.2, 7, and 10 for perfect sampling from attractive spin systems will also apply, for example, to generating random independent sets in a bipartite graph or random perfect matchings in a graph, since it is explained in [31] how such structures can be viewed as distributive lattices.

To begin the description of an attractive spin system, consider a finite set V of vertices, regarded as *sites*. A *configuration* $\mathbf{x} = (x_v;\ v \in V)$ assigns to each site v a *spin* x_v, either $+1$ ("up") or -1 ("down"). Partially order the set $\mathscr{S}$ of configurations by declaring that $\mathbf{x} \le \mathbf{y}$ exactly when $x_v \le y_v$ for every $v \in V$. Equivalently, we partially order the configurations by set inclusion, whereby a configuration is identified with the subset of sites where its spin is positive. Write $(\mathbf{x};\ x_v \leftarrow \sigma)$ for the configuration whose spin at v is σ and whose spins at other sites agree with those for $\mathbf{x}$.

Now let π be a probability distribution on $\mathscr{S}$ and let $\mathbf{x} \in \mathscr{S}$ with $\pi(\mathbf{x}) > 0$. Consider *updating* $\mathbf{x}$ at a specified site v according to π conditionally given the spins at every $w \neq v$. That is, consider resetting (if necessary) the spin at v to be $+1$ with probability

$$\begin{aligned} &\mathbf{P}_v(\mathbf{x}, (\mathbf{x};\ x_v \leftarrow +1)) \\ &\qquad := \pi(\mathbf{x}; x_v \leftarrow +1)/[\pi(\mathbf{x};\ x_v \leftarrow +1) + \pi(\mathbf{x}; x_v \leftarrow -1)] \end{aligned} \tag{2.1}$$

and -1 with the complementary probability $\mathbf{P}_v(\mathbf{x}, (\mathbf{x};\ x_v \leftarrow -1))$. Notice that these transition probabilities do not depend on x_v. If $\mathbf{P}_v(\mathbf{x}, (\mathbf{x}; x_v \leftarrow +1))$ is an increasing function of $\mathbf{x}$ (in the partial order), we say that π is *attractive*.

A famous example is the *Ising model*, whose definition (not in the greatest generality, but sufficient for this motivational introduction) is as follows. Let $G = (V, E)$ be a finite undirected graph. For $\mathbf{x} \in \mathscr{S}$, define

$$H(\mathbf{x}) := -\sum_{\{v,\, w\} \in E} x_v x_w.$$

For given $\theta \ge 0$, let

$$\pi(\mathbf{x}) := z_\theta^{-1} \exp(-\theta H(\mathbf{x})), \qquad \mathbf{x} \in \mathscr{S},$$

where $z_\theta = \sum_{\mathbf{x}} \exp(-\theta H(\mathbf{x}))$ is the normalizing constant. In the language of statistical mechanics, π is the *Gibbs distribution* (or *Gibbs state*) associated with the *Hamiltonian* H, the parameter θ is (proportional to) *inverse temper-*

ature, and z_θ is the *partition function*. A simple calculation shows that

$$\mathbf{P}_v(\mathbf{x}, (\mathbf{x};\ x_v \leftarrow +1)) = \left[1 + \exp\left(-2\theta \sum_{w:\,\{v,w\}\in E} x_w\right)\right]^{-1} \tag{2.2}$$

and hence π is attractive.

In applications, G is usually a rectangular or toroidal grid with $n := |V| = 64 \times 64$ or 128×128, for example. If $\theta \neq 0$, it is plainly infeasible to calculate the partition function or to sample in an elementary fashion from π. In such situations one may resort to *Markov chain Monte Carlo* (MCMC), as described in the next section.

3. Markov chain Monte Carlo. In the general setting of a probability distribution π on a set $\mathscr{S}$, the MCMC approach is to devise an ergodic (i.e., irreducible, aperiodic, and positive recurrent) Markov chain $\mathbf{X}$ with state space $\mathscr{S}$ and stationary distribution π. To obtain a single observation with approximate distribution π, one starts $\mathbf{X}$ in some conveniently generated distribution π_0 (for example, point mass at a convenient state $\mathbf{x}_0 \in \mathscr{S}$), runs it for "a long time" t (we shall have more to say about this presently), and uses $\mathbf{X}_t$ as the observation. Treating $\mathbf{X}_t$ as if it were exactly distributed according to π, one can continue running the chain to obtain a dependent sample from π, or be somewhat wasteful and repeatedly restart the process to obtain an i.i.d. sample. For a discussion of such methodological issues in MCMC, see Chapter 11, Section 4.3 in [3] and the references cited therein. We shall find it sufficiently interesting and challenging here to consider the problem of generating a sample of size one.

The Metropolis algorithm (e.g., [3], Chapters 11 and 12 or [14]) gives a quite general method for constructing such chains. In the setting of an attractive spin system, however, we consider an alternative, namely, the *Gibbs sampler* or *heat-bath algorithm*. Recall the definition (2.1) of the transition matrix (t.m.) $\mathbf{P}_v$ for updating of site v and note that it does not require calculating the partition function. Let $\mathbf{P} := n^{-1}\sum_{v\in V}\mathbf{P}_v$; that is, the chain $\mathbf{P}$ picks a random site and updates it. (This *random site updating* scheme has been chosen for definiteness. The common alternative $\mathbf{P} = \mathbf{P}_{v_1}\cdots\mathbf{P}_{v_n}$ known as *systematic site updating*, where $v_1, \dots, v_n$ is an arbitrary but fixed ordering of the sites, will be discussed in Section 11.1.) It is easy to see that $\mathbf{P}$ is ergodic (on $\mathscr{S} = \{-1, +1\}^V$, if $\pi(\mathbf{x}) > 0$ for all $\mathbf{x}$; see also Remark 7.1). This class of chains will serve as a running example throughout.

DEFINITION 3.1. A Gibbs sampler with random site updating on an attractive spin system will be called an *RSU chain*.

Given an ergodic Markov t.m. $\mathbf{P}$, the distribution $\pi_t = \pi_0\mathbf{P}^t$ of the chain at time t will approach π "in the long run." The key question, of course, is "How long is the long run?" Many heuristic diagnostic techniques have been devised to address this question for Markov chains, but the methods can encounter serious pitfalls [3], including metastability. Another approach is to bound rates of convergence to stationarity analytically. A tremendous amount of work has taken place along these lines in the last 15 years; see [15], [3], [33], [12] for discussion and myriad references. But deriving such bounds is sometimes, as described in [31], an "arduous undertaking."

A case in point is the Ising model. There are several MCMC algorithms for approximate sampling from π. With various restrictions on the underlying graph and the temperature, there are also accompanying "rapid mixing" bounds—bounds which imply that π_t is close to π when t is only polynomially large in $|V|$, which is the logarithm of the size of the state space. (See [14], [31], [33] for references.) However, the degrees and coefficients involved in these bounds are typically so large as to make the bounds ineffective for real-sized problems. There are, furthermore, many attractive spin systems (e.g., random independent sets in a bipartite graph) for which there are no known bounds sufficient even to imply rapid mixing.

An ideal remedy to MCMC difficulties would be a "self-verifying" algorithm that produces a *perfect* stationary observation in time on the order of some measure τ of the mixing time of the chain without using any explicit prior information about τ. At first thought, it may come as a surprise that a self-verifying algorithm exists for a class of chains that includes all RSU chains. We present such an algorithm in Section 7.

4. Monotonicity. Rapid stationary sampling for Markov chains with enormous state spaces, for example, the stochastic Ising model on a 64 × 64 grid, demands some prior information about the t.m. $\mathbf{P}$. Indeed, it is argued in [29] and in Chapter 9 of [3] that any "pure simulation" algorithm (i.e., $\mathbf{P}$ is unknown to the algorithm) for a generic irreducible N-state chain cannot terminate without sampling a transition from every state, so that the running time is at least of order N. However, efficient simulation is possible when, as is true for RSU chains, the chain is assumed monotone. Here is the definition.

DEFINITION 4.1. Let $\mathbf{P}$ be a t.m. on a partially ordered state space $(\mathscr{S}, \le)$. We say that $\mathbf{P}$ is *monotone* if $\mathbf{P}$ preserves the partial order, that is, if $\mathbf{P}(\mathbf{x}, \cdot) \le \mathbf{P}(\mathbf{y}, \cdot)$ stochastically whenever $\mathbf{x} \le \mathbf{y}$.

As background, recall the following definitions [35, 25]. Call a subset I of a partially ordered set (poset) $\mathscr{S}$ an *order ideal* or *down-set* if, whenever $\mathbf{x} \in I$ and $\mathbf{y} \le \mathbf{x}$, we have $\mathbf{y} \in I$. The set of all order ideals is denoted $J(\mathscr{S})$. Given two probability measures μ and ν on $\mathscr{S}$, one says that $\mu \le \nu$ *stochastically* if $\mu(I) \ge \nu(I)$ for all $I \in J(\mathscr{S})$. The product of monotone transition matrices is monotone.

Monotonicity of $\mathbf{P}$ is guaranteed when one can couple transitions using a monotone transition rule.

DEFINITION 4.2. A *monotone transition rule* for a t.m. $\mathbf{P}$ on a partially ordered state space $(\mathscr{S}, \le)$ is a measurable function $f\colon \mathscr{S} \times \mathscr{U} \to \mathscr{S}$, together with a random variable $\mathbf{U}$ taking values in an arbitrary probability space $\mathscr{U}$, such that (1) $f(\mathbf{x}, \mathbf{u}) \le f(\mathbf{y}, \mathbf{u})$ for all $\mathbf{u} \in \mathscr{U}$ whenever $\mathbf{x} \le \mathbf{y}$ and (2) $P(f(\mathbf{x}, \mathbf{U}) = \cdot) = \mathbf{P}(\mathbf{x}, \cdot)$ for all $\mathbf{x}$.

When a monotone transition rule exists, one can simultaneously generate transitions from various states in such a way as to maintain ordering relations for each realization.

Observe that each site update matrix $\mathbf{P}_v$ for an attractive spin system possesses a monotone transition rule; indeed, the coupling can be simply realized by letting U be uniformly distributed on $\mathscr{U} = [0, 1]$ and using

$$f_v(\mathbf{x}, u) = (\mathbf{x};\ x_v \leftarrow -1) \text{ or } (\mathbf{x};\ x_v \leftarrow +1) \qquad \text{according as } u \le \text{ or } > \mathbf{P}_v(\mathbf{x}, (\mathbf{x};\ x_v \leftarrow -1)). \tag{4.1}$$

The RSU chain also has a monotone transition rule: let $\mathscr{U} = \{1, \dots, n\} \times [0, 1]$; $\mathbf{U} = (U_1, U_2)$, where U_1 and U_2 are independent random variables uniformly distributed on $\{1, \dots, n\}$ and $[0, 1]$, respectively; and

$$f(\mathbf{x}, u_1, u_2) = f_{v_{u_1}}(\mathbf{x}, u_2). \tag{4.2}$$

REMARK 4.3. For definiteness and simplicity, we shall assume that each RSU chain transition generated using (4.2) is executed in time of constant order. This seems reasonable in the case of the stochastic Ising model on a toroidal grid, for example, since the computation of (2.2) involves a sum of four spin values, regardless of n or v or $\mathbf{x}$. Moreover, we shall adopt the assumption equally for all algorithms considered in this paper; thus comparisons between algorithms will be fair even if the assumption is violated.

Finally, we shall usually assume for simplicity that there exist elements $\hat{0}$ and $\hat{1}$ in $\mathscr{S}$ such that $\hat{0} \le x \le \hat{1}$ for all $x \in \mathscr{S}$; this condition is also met for attractive spin sytems.

DEFINITION 4.4. When a monotone transition rule and $\hat{0}$ and $\hat{1}$ exist for a given chain $\mathbf{P}$, we shall say that the *monotone case* obtains.

REMARK 4.5. One might conjecture that a monotone transition rule exists whenever $\mathbf{P}$ is monotone. Although this is true when the state space $\mathscr{S}$ is linearly ordered, joint work with Machida [17] has shown this to be false for general posets. So the monotone case entails a somewhat stronger condition than monotonicity of $\mathbf{P}$.

5. A backward coupling algorithm for the monotone case.

5.1. *Forward coupling.* Johnson [24] proposed testing for convergence of a MCMC algorithm using coupled sample paths. There are two underlying ideas involved. First, consider simultaneously running one copy of the specified Markov chain from each possible initial state, coalescing the various trajectories as they intersect and otherwise coupling the transitions in an arbitrary fashion. Stopping at the first time that all copies of the chain have coalesced, the effect of the initial state has worn off and equilibrium has been reached. Second, in the monotone case, if a monotone transition rule is used, then one need only run *two* copies of the chain, one each from the initial states $\hat{0}$ and $\hat{1}$. When these two trajectories have coalesced, so have all the others.

Although the first idea has obvious intuitive appeal, it is important to note that it does *not* lead to a *perfect* sampling scheme, as the following two examples demonstrate.

EXAMPLE 5.1. (As pointed out by a referee, this example was independently considered by Kendall [26].) Consider the following t.m. on the state space $\{0, 1\}$. From state 0 the chain moves to 0 or to 1, with probability 1/2 each. From state 1 the chain moves deterministically to 0. The stationary distribution is given by $\pi(0) = 2/3$, $\pi(1) = 1/3$. However, with probability 1, the two copies of the chain will first coalesce at state 0.

EXAMPLE 5.2. Here is an RSU chain counterexample. In the notation of Section 2, let G be a graph consisting of two sites, say v and w, and a single edge, and consider the Hamiltonian

$$H(\mathbf{x}) := -x_v x_w - x_v.$$

(This is the Ising model discussed in Section 2, modified to account for an external field which prefers a positive spin at site v.) Use random site updating $\mathbf{P} = (\mathbf{P}_v + \mathbf{P}_w)/2$. Through somewhat tedious calculations, one can find the distribution (p, say) of the configuration at first coalescence. The distributions p and π do not agree; for example,

$$p(\hat{0}) = (e^{\theta} + e^{-\theta})^{-2} - \frac{(e^{2\theta} - 1)^3}{4(e^{2\theta} + 1)^4} = \pi(\hat{0}) - \frac{(e^{2\theta} - 1)^3}{4(e^{2\theta} + 1)^4} < \pi(\hat{0}).$$

5.2. *The Propp–Wilson algorithm for the monotone case.* A modification of the coupling idea leads to an algorithm for *perfect* sampling from the stationary distribution π of an ergodic Markov chain. Working independently of Johnson, Propp and Wilson [31] devised a *backward coupling*, or *coupling from the past*, algorithm (call it PW) for efficient exact sampling in the monotone case. Their algorithm is simple to describe. For $t = 1, 2, 4, 8, \ldots,$ start two copies (say, $\mathbf{X}$ and $\mathbf{Y}$) of the chain at time $-t$, one in $\hat{0}$ and the other in $\hat{1}$. Run the two chains until time 0, coupling the transitions by means of a monotone transition rule f. If $\mathbf{X}_0 = \mathbf{Y}_0 = \mathbf{Z}$ (say), we say that the trajectories of $\mathbf{X}$ and $\mathbf{Y}$ have *coalesced* and return the value $\mathbf{Z}$. (Note that $\mathbf{Z}$ is *not necessarily* the state of first coalescence.) Otherwise, restart the procedure with the next value of t. The time and space requirements of this algorithm are discussed below. For Algorithm PW to work properly, it is essential, when the transitions from a given time s to time $s + 1$ are generated by means of $f(\mathbf{x}, \mathbf{U}_s)$, that the same value $\mathbf{U}_s$ be used in every iteration of the t-loop. If we assume that the algorithm terminates with probability 1 (as is true, e.g., in the attractive spin system case), then it is not hard to show that $\mathbf{Z} \sim \pi$ *exactly*. For this reason, Propp and Wilson describe their algorithm as one for exact sampling from π.

However, as Propp and Wilson ([31], Section 2.1) point out, the running time of the algorithm and $\mathbf{Z}$ are not independent. Thus occurrences of catastrophic system failures (more likely for long runs than for short), or early terminations of long runs by an impatient user, will result in a biased sample: observations that tend to take a long time to generate are underrepresented. [In connection with user impatience, notice that if no output has been generated by the end of a given iteration, the user knows that, due to time-doubling, the waiting time to termination (as measured by number of transitions) will be greater than the total time expired to that point.]

REMARK 5.3. We do not wish to overstate the importance of this bias, so several comments are in order here.

1. If suitable precautions are taken, a system crash need not lead to termination of a run. As a simple example of this, suppose that Algorithm PW is implemented using a seeded pseuodorandom number generator for an RSU chain, as discussed at the end of Section 6.2. Provided that the initial seed is written to disk before a run commences, the run can be started over after rebooting. Alternatively, one might wish to save partial results as the run progresses.

2. There would be little sense in stopping a run of Algorithm PW and beginning a new one, provided the implementation allows for pausing a run and resetting the time-doubling scheme. After all, for fixed $s \geq 0$, the conditional probability of coalescence from time $-(t+s)$ to time 0 given the simulation over the time interval $[-t, 0]$ is *minimized* (over all choices of t and all possible simulation results) when $t = 0$.

3. In light of comment 2, a somewhat more realistic scenario is that of a user who, for a predetermined length of time t, repeatedly runs the algorithm to collect observations from π, and then quits, perhaps before and perhaps after the run in progress at the planned quitting time is completed, obtaining the results $\mathbf{Z}_1, \mathbf{Z}_2, \ldots, \mathbf{Z}_{N(t)}$. Suppose that these observations will be used to estimate a functional $Eg(\mathbf{Z})$ of π (where $\mathbf{Z} \sim \pi$) by the sample mean $\bar{g} := N(t)^{-1} \sum_{i=1}^{N(t)} g(\mathbf{Z}_i)$. Curiously, when a run in progress is completed if and only if no other observations have been collected, it can be shown that the resulting estimator $\bar{g}$ is unbiased. On the other hand, if the observation in progress is always collected, then (due to length-biased sampling) the sample will be biased *in favor of* observations that take a long time to generate (in contrast to the bias against such observations caused by aborting long runs). In either case, it is important to realize that the *conditional* distribution of a fixed duration sample given its size is *not* that of an i.i.d. sample. Instead, a user would be advised to use training runs to estimate the distribution of running time and then sample a fixed number of observations rather than for a fixed computation time. For more on such use of training runs, see Section 6.1.

4. We have adopted a rather harsh standard of "bias" here and do not mean to single out the Propp–Wilson algorithm for criticism. Indeed, consider the standard algorithm for simulating a biased coin flip by a sequence of unbiased flips. To be more specific, suppose that independent random bits $b_1, b_2, \ldots$

are generated sequentially until it is determined whether or not the uniform random number $.b_1 b_2 \cdots$ (in binary notation) belongs to the interval $[0, 1/3]$. Given that a decision is reached after at most $2k$ (respectively, $2k + 1$) bits have been generated, the probability of an affirmative decision is $\frac{1}{3}$ [resp., $\frac{1}{3}(1 - 1/(2^{2k+1} - 1)) < \frac{1}{3}$]. So, for an impatient user, this algorithm is "biased" towards a negative decision. In light of our comments, perhaps "bias" should be read throughout as "potential for carelessly inappropriate use."

5. See also the cautionary note in the last paragraph of Section 7.3.

To provide a simplistic but analyzable model of user-impatience bias, we suppose that the user decides in advance to terminate a run of Algorithm PW if and when i_0 iterations have been completed without output. Remark 5.3 indicates that this model will provide a loose *upper bound* on the sort of bias incurred in practice.

Here is an exaggerated example of such bias, extracted from the discussion in Section 2.1 of [31]. Consider the Markov chain with states $\hat{0} = 0$, 1, and $\hat{1} = 2$ and monotone transition rule, using a uniform random variable U on $\mathscr{U} = [0, 1]$, given by $f(\mathbf{x}, u) = \max(\mathbf{x} - 1, 0)$ if $u \le 1/2$ and $f(\mathbf{x}, u) = \min(\mathbf{x} + 1, 2)$ if $u > 1/2$. Let $i_0 = 2$. If we use the Propp–Wilson algorithm to sample from π, but abort runs not completed in i_0 iterations, then (conditionally given completion) $\mathbf{Z} = 0, 1, 2$ with respective probabilities $1/2, 0, 1/2$, whereas $\pi = (1/3, 1/3, 1/3)$.

To obtain an example for RSU chains, consider again the set-up of Example 5.2, and let $i_0 = 2$. An easy calculation gives the probabilities of completion with $\mathbf{Z} = \hat{0}$ and completion with $\mathbf{Z} = \{v\}$ (= the configuration with spin $+1$ at v and -1 at w) as

$$\frac{e^{2\theta} + 1}{8(e^{4\theta} + 1)} \quad \text{and} \quad \frac{1}{4(e^{2\theta} + 1)},$$

respectively. The ratio of these probabilities (for $\{v\}$ to $\hat{0}$) is $2(e^{4\theta}+1)/(e^{2\theta}+1)^2$, which is larger than the corresponding ratio of 1 for the stationary probabilities.

In Sections 7 and 10 we shall present alternative algorithms which handle RSU chains (among others) and eliminate the user-impatience bias.

In passing, we offer some general remarks about perfect simulation in general and backward coupling in particular. See Chapter 4 in [3] and [26] for general discussion of perfect simulation and references to uses of backward coupling predating Algorithm PW. With a somewhat different formulation, backward coupling has also been developed and employed effectively by Foss [19] and by Borovkov and Foss [9, 10]. For infinite state space extensions and applications of Algorithm PW, see [30]. For a variant of backward coupling using regeneration times, see [20].

6. Analysis of the Propp–Wilson algorithm.

6.1. *Assessing the bias.* What is an upper bound on the user-impatience bias associated with Algorithm PW? To answer this question, consider the distributions

$$\sigma_t := \mathscr{L}(\mathbf{Z} \,|\, T \le t), \qquad t = 0, 1, 2, \ldots,$$

where T is the *backward coalescence time*, i.e., the smallest value of t such that the coalescence occurs by time 0 when the two copies of the chain are started at time $-t$. In passing, we record the simple fact, noted in Section 5.1 of [31], that T has marginally the same distribution as the *forward coalescence time*, that is, the time it takes transition-coupled trajectories started in $\hat{0}$ and $\hat{1}$ to first coalesce. Generalizing σ_t, for any random variable $\hat{T}$ let

$$\sigma(\hat{T}) \equiv \sigma(\mathscr{L}(\hat{T})) := \mathscr{L}(\mathbf{Z} \,|\, T \le \hat{T}).$$

If $\hat{T}$ is generated independent of the randomness used to generate $(T, \mathbf{Z})$, then $\sigma(\hat{T})$ is the mixture of the distributions σ_t with respect to the mixing measure $\mathscr{L}(\hat{T} \,|\, T \le \hat{T})$ on t. Proposition 6.2, a cousin to results in Section 5.1 of [31], bounds the variation distance

$$\|\sigma(\hat{T}) - \pi\| = \max_{\mathbf{A} \subseteq \mathscr{S}} |P(\mathbf{Z} \in \mathbf{A} \,|\, T \le \hat{T}) - \pi(\mathbf{A})|$$

from stationarity. We first state and prove a very general lemma, from which (since $\mathbf{Z} \sim \pi$ unconditionally) the proposition follows immediately, and then prove and discuss two useful consequences of the proposition.

LEMMA 6.1. *Let $(\Omega, \mathscr{A}, P)$ be any probability space and let $B \in \mathscr{A}$ with $P(B) > 0$. Let $\mathbf{Z}$ be any measurable mapping into any measurable space $(\Omega', \mathscr{A}')$. Consider the variation distance*

$$d := \|\mathscr{L}(\mathbf{Z}|B) - \mathscr{L}(\mathbf{Z})\| = \sup_{\mathbf{A} \in \mathscr{A}'} |P(\mathbf{Z} \in \mathbf{A} \,|\, B) - P(\mathbf{Z} \in \mathbf{A})|.$$

Then

$$d \le P(B^c)/P(B).$$

PROOF. For any $\mathbf{A} \in \mathscr{A}'$,

$$\begin{aligned} P(B)[P(\mathbf{Z} \in \mathbf{A} \,|\, B) - P(\mathbf{Z} \in \mathbf{A})] &= P(B \cap \{\mathbf{Z} \in \mathbf{A}\}) - P(B)P(\mathbf{Z} \in \mathbf{A}) \\ &= P(B^c)P(\mathbf{Z} \in \mathbf{A}) - P(B^c \cap \{\mathbf{Z} \in \mathbf{A}\}), \end{aligned}$$

the absolute value of which does not exceed $P(B^c)$. □

PROPOSITION 6.2. *In the setting described above,*

$$\|\sigma(\hat{T}) - \pi\| \le \frac{P(T > \hat{T})}{1 - P(T > \hat{T})}.$$

COROLLARY 6.3. *Consider again the set-up of Proposition* 6.2. *If $T, T_1, \ldots T_r$ are independent random variables, each distributed as the forward coalescence* *time, and* $\hat{T} := T_1 + \cdots + T_r$, *then*

$$\|\sigma(\hat{T}) - \pi\| \le \frac{1}{2^r - 1}.$$

PROOF. Using the proposition, we need only show that $P(T > \hat{T}) \le 2^{-r}$. But this is done in Section 5.1 in [31] and is easy: condition on the values of $T_1, \dots, T_r$, use the easily established submultiplicativity of the tails of $\mathscr{L}(T)$ (Lemma 5 in [31]), and observe $P(T > T_1) \le 1/2$. □

Without going into details, we remark that an alternative to the corollary can be obtained by considering $\max\{T_1, \dots, T_r\}$ in place of $\hat{T}$.

Proposition 6.2 and Corollary 6.3 suggest methods for controlling the user impatience bias, as measured by the variation distance $b(\hat{T}) := \|\sigma(\hat{T}) - \pi\|$. Suppose first that $\hat{T} \equiv t$. There is no need to do any backward coupling. Indeed, suppose one simply repeatedly runs coupled trajectories forward from initial states $\hat{0}$ and $\hat{1}$, each for duration t, until such a run produces coalescence, and then outputs the terminal state of that run. It follows from Proposition 6.2 that the variation distance between the law of the eventual output and the desired π is $b(t) \le P(T > t)/[1 - P(T > t)]$.

Of course this information is useless except in those rare instances when one has at hand a bound on the tails of $\mathscr{L}(T)$. Furthermore, in those cases one can do slightly better than Proposition 6.2 with an even simpler algorithm, as follows. It is clear (see [31], Theorem 4) that the coupling inequalities

$$d(t) \le \bar{d}(t) \le P(T > t),$$

hold, where

$$d(t) := \max_{\mathbf{x} \in \mathscr{S}} \|\mathbf{P}^t(\mathbf{x}, \cdot) - \pi\| \quad \text{and} \quad \bar{d}(t) := \max_{\mathbf{x}, \mathbf{y} \in \mathscr{S}} \|\mathbf{P}^t(\mathbf{x}, \cdot) - \mathbf{P}^t(\mathbf{y}, \cdot)\|. \tag{6.1}$$

So one need only run a single trajectory forward for t steps from an arbitrarily chosen initial state, and the variation distance between the law of the terminal state and π is at most $P(T > t)$.

Typically, one will need to estimate the distribution of T statistically, and then similar comments apply to the bound of Corollary 6.3. The upshot is that one can begin by obtaining a "training sample" $T_1, \dots, T_r$ of r forward coalescence times, let $\hat{T} = T_1 + \cdots + T_r$, and then run a single trajectory forward for $\hat{T}$ steps from an arbitrary initial state, resulting in a variation distance of at most 2^{-r}.

While it may seem that we have provided a method for controlling bias, there are several inherent drawbacks.

1. The need for training runs increases the overall running time of the sampling algorithm. The factor of increase grows without bound as the variation distance tolerance shrinks to 0.
2. The user must commit the resources in advance to completing the r training runs; if an impatient user terminates one of these runs early, then the

analysis provided by Corollary 6.3 is rendered invalid. But our demand is ridiculous, as the user need only commit to a *single* run of Algorithm PW to obtain a completely unbiased observation from π!

3. In our discussion, only bounds on *absolute* probability errors have been given. No such bounds exist for *relative* errors, as the following example demonstrates.

All these difficulties are overcome by the algorithm proposed in Section 7, since it is not subject to user-impatience bias.

EXAMPLE 6.4. Let $\mathscr{S}$ be the linearly ordered state space $\{0, 1, \dots, d\}$, with $d \ge 4$ assumed for later convenience. Let $\mathbf{P}_{\mathrm{RW}}$ govern the usual simple symmetric random walk, with holding probability $1/2$ at $\hat{0} = 0$ and at $\hat{1} = d$. Let $\mathbf{P}_{\mathrm{jump}}$ govern the chain with absorbing state d, which with probability 1 is reached at one step from any other state. Then

$$\mathbf{P}_{\mathrm{mix}} := \frac{1}{d}\mathbf{P}_{\mathrm{RW}} + \left(1 - \frac{1}{d}\right)\mathbf{P}_{\mathrm{jump}}$$

inherits ergodicity from $\mathbf{P}_{\mathrm{RW}}$ and possession of a monotone transition rule from $\mathbf{P}_{\mathrm{RW}}$ and $\mathbf{P}_{\mathrm{jump}}$. Let $\mathbf{P} = \mathbf{P}_{\mathrm{mix}}$. Then $P(T > 1) = 1/d$ and σ_1 is unit mass δ_d at d, so Proposition 6.2 with $\hat{T} \equiv 1$ implies

$$\|\delta_d - \pi\| \le \frac{1}{d-1},$$

which is small when d is large. (It is also clear directly that $\|\delta_d - \pi\| \le 1/d$.) But approximation of $\pi(\mathbf{x}) > 0$ for any given $\mathbf{x} < d$ by $\delta_d(\mathbf{x}) = 0$ incurs a relative error of 1.

REMARK 6.5. In regard to drawback 2, amortization can provide some relief. That is, one might hope that commitment to completion of a single small sample from $\mathscr{L}(T)$ would allow for a large sample of observations from π that is relatively free of variation bias. Indeed, suppose that we generate a training sample $T_1, \dots, T_r$ from $\mathscr{L}(T)$, compute $\hat{T} = T_1 + \cdots + T_r$, and then obtain a sample $\mathbf{Z}_1, \dots, \mathbf{Z}_N$ by running N duration-$\hat{T}$ trajectories from arbitrarily chosen initial states. Then the variation distance between $\mathscr{L}(\mathbf{Z}_1, \dots, \mathbf{Z}_n)$ and perfection $(\pi \times \cdots \times \pi)$ is bounded by

$$P(\max(T'_1, \dots, T'_N) > \hat{T})$$

where $T_1, \dots, T_r, T'_1, \dots, T'_N$ are all independent copies of T. But

$$\begin{aligned} P\big(\max(T'_1, \dots, T'_N) \le \hat{T} \mid T_1 = t_1, \dots, T_r = t_r\big) &= \big[P(T'_1 \le t_1 + \cdots + t_r)\big]^N \\ &\ge \left[1 - \prod_{i=1}^{r} P(T'_1 > t_i)\right]^N \\ &\ge 1 - N \prod_{i=1}^{r} P(T'_1 > t_i), \end{aligned}$$

from which it follows easily that

$$P\bigl(\max(T'_1, \dots, T'_N) > \hat{T}\bigr) \le N2^{-r}.$$

Therefore, to guarantee that the sample $\mathbf{Z}_1, \dots, \mathbf{Z}_N$ has variation bias no greater than ε, we need only choose $r \ge \lg(N/\varepsilon)$. Thus, for example, to double the desired sample size for a given tolerance ε, one need not double the required training size, but rather only increase it by 1.

6.2. *Time and space requirements.* We conclude Section 6 by treating the time and space requirements of the Propp–Wilson algorithm. The running time was treated in [31], to which we refer the reader for further details, but the space analysis is new. As defined in Chapter 4 of [3] in the case of reversible Markov chains, let τ_1 denote the mixing time parameter

$$\tau_1 := \min\bigl\{t > 0\colon \bar{d}(t) \le e^{-1}\bigr\}, \tag{6.2}$$

called the *variation threshold*, of a Markov chain $\mathbf{P}$, where the maximum variation distance $\bar{d}(t)$ is defined at (6.1). Then, when the algorithm is applied to any monotone case chain $\mathbf{P}$ on a poset $\mathscr{S}$, the running time is linear in $T\Delta$, where Δ is the time required to perform a single $\mathbf{P}$-step and T is the backward coalescence time as defined in Section 6.1. Propp and Wilson show that

$$ET \le 2\tau_1(1 + \ln l), \tag{6.3}$$

where l is the length of the longest chain in $\mathscr{S}$. For an RSU chain [so that $\Delta = \Theta(1)$ (see Remark 4.3) and $l = n$], this implies that the expected run time is $O(\tau_1 \log n)$.

We analyze the memory needs only in the case of RSU chains. The space required depends on the source of the random variables used to generate the coupled transitions. Suppose first that the source is actual (unreproducible) randomness. As mentioned above, the values $\mathbf{U} = (U_1, U_2) \in \{1, \dots, n\} \times [0, 1]$ used for an update at one iteration need to be reused in subsequent iterations. Hence the space requirement (in bits) for the U_1-values is $\Theta(T \log n)$, which has expectation $O(\tau_1 (\log n)^2)$; and $\Theta(T)$, which has expectation $O(\tau_1 \log n)$, is the number of uniform U_2-values that must also be stored. Note, however, that only $\Theta(n + \log T)$ [with expectation $O(n + \log \tau)$] bits of read/write memory are needed; the $\mathbf{U}$-values can be read from read-only memory, while the additional $\Theta(\log T)$ term accounts for the algorithm's need to keep track of time.

Wilson [38] has observed that, by an imputation trick, it suffices to store ternary digits rather than U_2-values, as follows. Let $i \ge 0$. During the $(i + 1)$st iteration (i.e., starting at time -2^i), store a ternary digit d for each site update until time 0. If both copies of the chain spin the site to $+1$ or both spin to -1, store $d = +1$ or $d = -1$, respectively. If the $\hat{1}$-copy of the chain spins the site to $+1$ and the $\hat{0}$-copy spins it to -1, store $d = 0$. How to spin the sites is determined as follows. Over the interval $[-2^i, -2^{i-1})$, generate fresh uniform values from $[0, 1]$ and use the transition rule. Over the interval $[-2^{i-1}, 0)$, the stored digits are used to redetermine the trajectories from the previous iteration and to determine the trajectories for the present iteration, and then are overwritten, as follows. If $d = +1$ or $d = -1$ is stored, spin the site in both copies to $+1$ or -1, respectively, in both the present and previous iterations' trajectories, and leave d unchanged. If $d = 0$ is stored, the previous iteration's trajectories are extended by the appropriate spinning. Further, we know that the corresponding uniform from the previous iteration had a value u_2 in the interval $(\mathbf{P}_v(\mathbf{y}, (\mathbf{y}; y_v \leftarrow -1)), \mathbf{P}_v(\mathbf{x}, (\mathbf{x}; x_v \leftarrow -1))]$, where $\mathbf{x}$ and $\mathbf{y}$ are the respective values of the copies of the chain started at time -2^{i-1} in states $\hat{0}$ and $\hat{1}$. The value u_2 can be imputed by generating a fresh uniform value in this interval. Use this value and the transition rule to determine the spins for the two current copies of the chain, and store the appropriate value of d. With the trick we have described, the total memory requirement for the algorithm is $\Theta(T \log n + n)$ bits, of which $\Theta(T + n)$ must be read/write memory; the corresponding expected values are $O(\tau_1 (\log n)^2 + n)$ and $O(\tau_1 \log n + n)$. Of course, for virtually all real applications, the first of the two terms will predominate in each case. Although this trick reduces the *total* memory requirement, if (as is quite typically the case) read-only memory is much more abundantly available than read/write memory and T is of larger order of magnitude than n, then our analysis shows that it would be better not to use the trick.

Now suppose instead that a typical *seeded pseudorandom number generator* is used; that is, we assume that each value of $\mathbf{U}$ is determined (in time of constant order) from a pseudorandom number s_j generated by applying a deterministic function r to the preceding pseudorandom number s_{j-1} generated, where the process is begun with some specified seed s_0. Storage can then be greatly reduced, at the expense of merely a constant factor in running time, by storing only the initial seed s_0: whenever time -2^i is reached, the algorithm simply recomputes the seed $s_{2^{i-1}}$ required at that time via $s_{2^{i-1}} = r^{2^{i-1}}(s_0)$, where r^j denotes the jth iterate of r. (For $i = 0$, the required seed is s_0.) The space requirement is $\Theta(n)$ (needed to keep track of two configurations), plus the memory requirement m for the seed s_0. One might reasonably take m to be $\Theta(\log T)$ in order to get good performance from the generator. The total expected space requirement is then $O(n + \log \tau_1)$. We may reasonably assume that $\log \tau_1 = O(n)$, and then the space needed is $O(n)$. Indeed, if τ_1 is not of order at most *polynomial* in n, then *no* feasible MCMC algorithm can hope to approximate the stationary distribution when n is very large. So we may assume that $\tau_1 = O(n^\alpha)$ for some $\alpha < \infty$, although the governing α may be application-dependent.

7. A new algorithm based on rejection sampling.

7.1. *Background.* To set the stage for the new algorithm, we review some background material. Let $\mathscr{S}$ be a finite poset, and consider probability measures μ and ν on $\mathscr{S}$. It is well known (e.g., Theorem 1 in [25]; see also [36]) that $\mu \le \nu$ (stochastically) is equivalent to the existence of an *upward kernel* $\mathbf{K}$ such that $\nu = \mu \mathbf{K}$, that is, of a Markov t.m. $\mathbf{K}$ on $\mathscr{S}$ such that (1) for all $\mathbf{x} \in \mathscr{S}$, the measure $\mathbf{K}(\mathbf{x}, \cdot)$ is supported on $\{\mathbf{y} \in \mathscr{S}\colon \mathbf{y} \ge \mathbf{x}\}$, and (2) for all

$\mathbf{y} \in \mathscr{S}$, we have $\nu(\mathbf{y}) = \sum_{\mathbf{x}\in\mathscr{S}} \mu(\mathbf{x})\mathbf{K}(\mathbf{x},\mathbf{y})$. Also, recall the definition $\tilde{\mathbf{P}}(\mathbf{x},\mathbf{y}) := \pi(\mathbf{y})\mathbf{P}(\mathbf{y},\mathbf{x})/\pi(\mathbf{x})$ of the *time-reversal* of a Markov t.m. $\mathbf{P}$ with stationary distribution π. For example, it is evident from (2.1) that each $\mathbf{P}_v$ is reversible with respect to π (i.e., $\tilde{\mathbf{P}}_v = \mathbf{P}_v$); hence so is the RSU chain $\mathbf{P} = n^{-1}\sum_v \mathbf{P}_v$. The algorithm we shall present does *not* require reversibility of $\mathbf{P}$.

REMARK 7.1. For simplicity, we have supposed that the t.m. $\mathbf{P}$ of interest is ergodic, so that $\pi(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathscr{S}$. When, as discussed in Section 2, $\mathscr{S}$ is the Boolean algebra of all subsets of a finite poset V and $\mathbf{P}$ is the RSU chain with long-run distribution π that is uniform over the collection $J(V)$ of order ideals of V, the assumption of ergodicity is violated but can be restored by a suitable restriction of attention to $\mathscr{S}' = J(V)$. We omit the details.

We assume that $\tilde{\mathbf{P}}$ is monotone on $\mathscr{S}$ and that $\mathscr{S}$ possesses $\hat{0}$ and $\hat{1}$, but we do not require that a monotone transition rule exist for $\tilde{\mathbf{P}}$. For each $\mathbf{x} \in \mathscr{S}$, let $\mu_{\mathbf{x}} := \tilde{\mathbf{P}}(\mathbf{x},\cdot)$. Monotonicity of $\tilde{\mathbf{P}}$ is equivalent to the assumption that when $\mathbf{x} \le \mathbf{y}$ there exists an upward kernel $\mathbf{K}_{\mathbf{x},\mathbf{y}} \equiv \mathbf{K}_{\mathbf{x},\mathbf{y}}(\cdot,\cdot)$ such that $\mu_{\mathbf{y}} = \mu_{\mathbf{x}}\mathbf{K}_{\mathbf{x},\mathbf{y}}$. We assume that the user can simulate transitions from the measure $\mathbf{K}_{\mathbf{x},\mathbf{y}}(\mathbf{x}',\cdot)$ whenever $\mathbf{x} \le \mathbf{y}$ and $\tilde{\mathbf{P}}(\mathbf{x},\mathbf{x}') > 0$.

Suppose, for example, that a monotone transition rule $\tilde{f}$ *does* exist for $\tilde{\mathbf{P}}$, that is, that the monotone case obtains for $\tilde{\mathbf{P}}$ in the sense of Definition 4.4. It is then easy to check that one can use

$$\mathbf{K}_{\mathbf{x},\mathbf{y}}(\mathbf{x}',\mathbf{y}') := P(\tilde{f}(\mathbf{y},\mathbf{U}) = \mathbf{y}' \mid \tilde{f}(\mathbf{x},\mathbf{U}) = \mathbf{x}'), \qquad \mathbf{y}' \in \mathscr{S} \tag{7.1}$$

(when $\mathbf{x} \le \mathbf{y}$ and $\tilde{\mathbf{P}}(\mathbf{x},\mathbf{x}') > 0$) for the upward kernels. In particular, if one can generate a random variable $\hat{\mathbf{U}}$ (say) whose distribution is the conditional distribution of $\mathbf{U}$ given $\tilde{f}(\mathbf{x},\mathbf{U}) = \mathbf{x}'$, then one can simulate an observation $\mathbf{Y}'$ from $\mathbf{K}_{\mathbf{x},\mathbf{y}}(\mathbf{x}',\cdot)$ by setting $\mathbf{Y}' = \tilde{f}(\mathbf{y},\hat{\mathbf{U}})$.

EXAMPLE 7.2. When we specialize to RSU chains, a straightforward calculation establishes the validity of the following simple method for generating $\mathbf{Y}'$ (as in the preceding paragraph) when $\mathbf{x}' \ne \mathbf{x}$ (so that $\mathbf{x}$ and $\mathbf{x}'$ disagree at a unique site v). If $x_v = -1$ and $\mathbf{x}' = (\mathbf{x};\ x_v \leftarrow +1)$, then set $\mathbf{Y}' = (\mathbf{y};\ y_v \leftarrow +1)$ (deterministically). If $x_v = +1$ and $\mathbf{x}' = (\mathbf{x};\ x_v \leftarrow -1)$, then set $\mathbf{Y}'$ to $(\mathbf{y};\ y_v \leftarrow -1)$ with probability $\mathbf{P}_v(\mathbf{y},(\mathbf{y};\ y_v \leftarrow -1))/\mathbf{P}_v(\mathbf{x},(\mathbf{x};\ x_v \leftarrow -1))$ and to $(\mathbf{y};\ y_v \leftarrow +1)$ with the complementary probability. A method for generating $\mathbf{Y}'$ in the more problematic case $\mathbf{x}' = \mathbf{x}$ will be discussed in Section 10.1.

7.2. *The algorithm in the general monotone setting.* Let $\mathbf{P}$ be an ergodic t.m. with monotone time-reversal $\tilde{\mathbf{P}}$ on a poset $\mathscr{S}$ possessing a unique maximum element $\hat{0}$ and a unique maximum element $\hat{1}$. Let $\mathbf{K}_{\mathbf{x},\mathbf{y}}$ be upward kernels for $\tilde{\mathbf{P}}$, as discussed in Section 7.1. The proposed new algorithm to output a stationary observation runs as follows. Independently for $t = 1, 2, 4, 8, \ldots$, perform the following routine; stop when output is first obtained. For the first phase of the routine, run the $\mathbf{P}$-chain $\mathbf{X}$ for t steps starting from state $\hat{0}$, recording the trajectory $(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_t)$. Suppose that $\mathbf{X}_t = \mathbf{z}$. For the second phase, regard $(\mathbf{X}_t, \mathbf{X}_{t-1}, \ldots, \mathbf{X}_0)$ as a trajectory $(\tilde{\mathbf{X}}_0, \tilde{\mathbf{X}}_1, \ldots, \tilde{\mathbf{X}}_t)$ from the time-reversed chain $\tilde{\mathbf{X}}$. Then, in synchrony with the evolution of $\tilde{\mathbf{X}}$, build a second trajectory $\tilde{\mathbf{Y}}$, which starts at state $\hat{1}$ and evolves according to the following one-step rule. Corresponding to each observed transition, say from $\mathbf{x}$ to $\mathbf{x}'$, of $\tilde{\mathbf{X}}$, let $\tilde{\mathbf{Y}}$ choose the transition from its present state $\mathbf{y}$ as $\mathbf{y}'$ with probability $\mathbf{K}_{\mathbf{x},\mathbf{y}}(\mathbf{x}', \mathbf{y}')$, $\mathbf{y}' \in \mathscr{S}$. (Notice that this construction gives $\tilde{\mathbf{X}}_s \le \tilde{\mathbf{Y}}_s$ for $0 \le s \le t$.) If $\tilde{\mathbf{Y}}_t = \hat{0}$, output $\mathbf{z}$; otherwise, erase all information and go on to the next value of t.

REMARK 7.3. The new algorithm bears superficial resemblance to techniques of Besag and Clifford [6, 7] in a different setting, but we are unaware of any useful connections.

7.3. *Proof of correctness.* Why does the algorithm presented in Section 7.2 work? The basic reason is that each iteration of the two-phase routine is an implementation of rejection sampling (cf., e.g., Chapter 10, Section 2.2 of [32]). We use an observation from $\mathbf{P}^t(\hat{0}, \cdot)$ to simulate one from π. In order to effect this, one produces an upper bound c on the ratio $\pi(\mathbf{z})/\mathbf{P}^t(\hat{0}, \mathbf{z})$ [subject to $\mathbf{P}^t(\hat{0}, \mathbf{z}) > 0$], generates $\mathbf{z}$ according to $\mathbf{P}^t(\hat{0}, \cdot)$, and (conditionally) accepts $\mathbf{z}$ as an observation from π with probability $c^{-1}\pi(\mathbf{z})/\mathbf{P}^t(\hat{0}, \mathbf{z})$. The unconditional probability of acceptance is then $1/c$.

Now

$$\pi(\mathbf{z})/\mathbf{P}^t(\hat{0}, \mathbf{z}) = \pi(\hat{0})/\tilde{\mathbf{P}}^t(\mathbf{z}, \hat{0}), \tag{7.2}$$

so the monotonicity of $\tilde{\mathbf{P}}$ implies that we may choose $c = \pi(\hat{0})/\tilde{\mathbf{P}}^t(\hat{1}, \hat{0})$ and thus accept $\mathbf{z}$ with probability

$$\frac{\tilde{\mathbf{P}}^t(\hat{1}, \hat{0})}{\pi(\hat{0})} \times \frac{\pi(\mathbf{z})}{\mathbf{P}^t(\hat{0}, \mathbf{z})} = \frac{\tilde{\mathbf{P}}^t(\hat{1}, \hat{0})}{\tilde{\mathbf{P}}^t(\mathbf{z}, \hat{0})}.$$

The first phase of the routine samples from $\mathbf{P}^t(\hat{0}, \cdot)$, and so the question in designing the algorithm becomes how to engineer a coin-flip with probability $\tilde{\mathbf{P}}^t(\hat{1}, \hat{0})/\tilde{\mathbf{P}}^t(\mathbf{z}, \hat{0})$ of heads. But, conditionally given that $\mathbf{X}$ starts at $\hat{0}$ and ends at $\mathbf{z}$, the reverse trajectory $\tilde{\mathbf{X}} = (\tilde{\mathbf{X}}_0, \ldots, \tilde{\mathbf{X}}_t) = (\mathbf{X}_t, \ldots, \mathbf{X}_0)$ has the distribution of a $\tilde{\mathbf{P}}$-trajectory conditioned to start at $\mathbf{z}$ and end at $\hat{0}$. Moreover, the second phase is designed precisely so that the probability of acceptance is the desired $\tilde{\mathbf{P}}^t(\hat{1}, \hat{0})/\tilde{\mathbf{P}}^t(\mathbf{z}, \hat{0})$.

LEMMA 7.4.

$$P\bigl(\tilde{\mathbf{Y}}_t = \hat{0} \,\big|\, \tilde{\mathbf{X}}_0 = \mathbf{z},\ \tilde{\mathbf{X}}_t = \hat{0},\ \tilde{\mathbf{Y}}_0 = \hat{1}\bigr) = \frac{\tilde{\mathbf{P}}^t(\hat{1}, \hat{0})}{\tilde{\mathbf{P}}^t(\mathbf{z}, \hat{0})}.$$

PROOF. Consider a modified situation where the trajectory $\tilde{\mathbf{X}} = (\tilde{\mathbf{X}}_0, \ldots, \tilde{\mathbf{X}}_t)$ is again a $\tilde{\mathbf{P}}$-trajectory, but no longer conditioned to start at $\mathbf{z}$ nor to end at

$\hat{0}$. We can again build a second (coupled) trajectory $\tilde{\mathbf{Y}}$ starting at state $\hat{1}$ and evolving as in the second phase of the algorithm: if $\tilde{\mathbf{X}}$ moves from $\mathbf{x}$ to $\mathbf{x}'$, then $\tilde{\mathbf{Y}}$ moves from $\mathbf{y}$ to $\mathbf{y}'$ with probability $\mathbf{K}_{\mathbf{x},\mathbf{y}}(\mathbf{x}', \mathbf{y}')$. As before, we will have $\tilde{\mathbf{X}}_s \le \tilde{\mathbf{Y}}_s$ for $0 \le s \le t$; in particular, $\tilde{\mathbf{Y}}_t = \hat{0}$ entails $\tilde{\mathbf{X}} = \hat{0}$. Moreover, in this modified setting it is easy to verify that $\tilde{\mathbf{X}}_0$ and $\tilde{\mathbf{Y}} = (\tilde{\mathbf{Y}}_0, \dots, \tilde{\mathbf{Y}}_t)$ are independent and that $\tilde{\mathbf{Y}}$ is (marginally) a Markov chain with t.m. $\tilde{\mathbf{P}}$.

Thus, returning to the setting of the algorithm, we have

$$\begin{aligned} P(\tilde{\mathbf{Y}}_t = \hat{0} \,|\, \tilde{\mathbf{X}}_0 = \mathbf{z}, \tilde{\mathbf{X}}_t = \hat{0}, \tilde{\mathbf{Y}}_0 = \hat{1}) &= \frac{P(\tilde{\mathbf{X}}_t = \hat{0}, \tilde{\mathbf{Y}}_t = \hat{0} \,|\, \tilde{\mathbf{X}}_0 = \mathbf{z}, \tilde{\mathbf{Y}}_0 = \hat{1})}{P(\tilde{\mathbf{X}}_t = \hat{0} \,|\, \tilde{\mathbf{X}}_0 = \mathbf{z}, \tilde{\mathbf{Y}}_0 = \hat{1})} \\ &= \frac{P(\tilde{\mathbf{Y}}_t = \hat{0} \,|\, \tilde{\mathbf{X}}_0 = \mathbf{z}, \tilde{\mathbf{Y}}_0 = \hat{1})}{\tilde{\mathbf{P}}^t(\mathbf{z}, \hat{0})} = \frac{\tilde{\mathbf{P}}^t(\hat{1}, \hat{0})}{\tilde{\mathbf{P}}^t(\mathbf{z}, \hat{0})}, \end{aligned}$$

as desired. □

The unconditional probability of acceptance is

$$c^{-1} = \frac{\tilde{\mathbf{P}}^t(\hat{1}, \hat{0})}{\pi(\hat{0})} = \frac{\mathbf{P}^t(\hat{0}, \hat{1})}{\pi(\hat{1})}, \tag{7.3}$$

which (by ergodicity) converges to 1 as $t \to \infty$. It follows that the conditional probability that the algorithm terminates at the ith iteration given failure to do so previously converges to 1 as i becomes large. A fortiori, the algorithm terminates with probability 1.

The claim of unbiasedness with respect to user impatience follows easily from the very nature of rejection sampling together with the fact that all information is erased after each iteration. However, we warn that for this claim to be valid, user impatience must be expressed in terms of number of transitions generated, not real computation time. Otherwise, care must be taken to program each transition to take the same (worst-case) amount of time. Wilson [38] points out that this will cause a crippling slowdown for chains with highly variable computation time for transitions; see Section 4.2 of [31] for such an example.

8. Performance of the new algorithm.

From the description of the new algorithm presented in Section 7.2, it is clear that one can analyze both the time and space requirements by studying the distribution of F, defined to be the number of forward $\mathbf{P}$-transitions generated. The algorithm will be fine-tuned for RSU chains in Section 10, and in Section 11 we compare the performance of the new algorithm with that of Algorithm PW, both generally and also specifically for RSU chains.

If $I \ge 1$ denotes the (random) number of iterations used by the algorithm, then

$$F = \sum_{h=0}^{I-1} 2^h = 2^I - 1$$

and we have already observed [see (7.3)] that

$$P(I > i | I > i-1) = \mathrm{sep}_{\hat{0}}(2^{i-1}), \qquad i \geq 1,$$

where

$$0 \leq \mathrm{sep}_{\hat{0}}(t) := 1 - \frac{\mathbf{P}^t(\hat{0}, \hat{1})}{\pi(\hat{1})} = \max_{\mathbf{y} \in \mathscr{S}} \left[1 - \frac{\mathbf{P}^t(\hat{0}, \mathbf{y})}{\pi(\mathbf{y})} \right] \leq 1, \qquad t \geq 0 \tag{8.1}$$

is recognized as the *separation* [3, 13] at time t for the $\mathbf{P}$-chain started in the state $\hat{0}$. The last equality in (8.1) follows from (7.2) and the monotonicity of $\tilde{\mathbf{P}}^t$. Thus F takes on values $2^i - 1$ with respective probabilities $P(I = i)$, $i \geq 1$, and it follows that

$$EF = \sum_{i=1}^{\infty} P(I = i) \sum_{h=0}^{i-1} 2^h = \sum_{h=0}^{\infty} 2^h P(I > h) = \sum_{h=0}^{\infty} 2^h \prod_{g=0}^{h-1} \mathrm{sep}_{\hat{0}}(2^g), \tag{8.2}$$

$$P(F > 2^h - 1) = P(I > h) = \prod_{g=0}^{h-1} \mathrm{sep}_{\hat{0}}(2^g), \qquad h \geq 0. \tag{8.3}$$

Let

$$\mathrm{sep}(t) := \max_{\mathbf{x}, \mathbf{y} \in \mathscr{S}} \left[1 - \frac{\mathbf{P}^t(\mathbf{x}, \mathbf{y})}{\pi(\mathbf{y})} \right] \tag{8.4}$$

be the separation maximized over choice of initial state. It is elementary and well known that for any Markov chain, $\mathrm{sep}(\cdot)$, like $\bar{d}(\cdot)$, is submultiplicative. As defined in Chapter 4 of [3] in the case of reversible Markov chains, let

$$\tau_1^{(1)} := \min\{t > 0 \colon \mathrm{sep}(t) \leq e^{-1}\}. \tag{8.5}$$

This mixing time parameter, called the *separation threshold*, is compared with τ_1 of (6.2) in Section 11.1 below.

The following theorem briefly summarizes the performance of our algorithm.

THEOREM 8.1. *Let F denote the* (*random*) *number of* $\mathbf{P}$*-transitions generated by the algorithm in Section* 7.2. *Then $EF = O(\tau_1^{(1)})$, where $\tau_1^{(1)}$ is the mixing time parameter given by* (8.5) *and* (8.4).

The next proposition, together with submultiplicativity of $\mathrm{sep}(\cdot)$, suffices to establish Theorem 8.1. The second conclusion asserts that the right tail of F decays at least geometrically quickly.

PROPOSITION 8.2. *Let*

$$\tilde{\tau} := \min\{t > 0 \colon \mathrm{sep}(t) \leq 1/2\} \quad \textit{and} \quad \tilde{h} := \lceil \lg \tilde{\tau} \rceil .$$

Then

$$EF \leq 8\tilde{\tau} \quad \textit{and} \quad P\big(F > 2^{\tilde{h}+h} - 1\big) \leq 2^{-(2^h - 1)} \qquad \textit{for } h \geq 0.$$

PROOF. To bound the last expression in (8.2), we proceed as in [37]. Observe that $\tilde{h}$ is the smallest integer h with $\text{sep}(2^h) \le 1/2$. Then

$$\begin{aligned} EF &\le \sum_{h=0}^{\tilde{h}} 2^h + \sum_{h=\tilde{h}+1}^{\infty} 2^h \prod_{g=\tilde{h}}^{h-1} \text{sep}(2^g) = 2^{\tilde{h}+1} - 1 + \sum_{h=\tilde{h}+1}^{\infty} 2^h \prod_{g=0}^{h-\tilde{h}-1} \text{sep}(2^{\tilde{h}+g}) \\ &\le 2^{\tilde{h}+1} + \sum_{h=\tilde{h}+1}^{\infty} 2^h \prod_{g=0}^{h-\tilde{h}-1} [\text{sep}(2^{\tilde{h}})]^{2^g} \qquad \text{(by submultiplicativity)} \\ &\le 2^{\tilde{h}+1} + \sum_{h=\tilde{h}+1}^{\infty} 2^h \prod_{g=0}^{h-\tilde{h}-1} 2^{-2^g} = 2^{\tilde{h}+1} + \sum_{h=\tilde{h}+1}^{\infty} 2^h 2^{-(2^{h-\tilde{h}}-1)} \\ &= 2^{\tilde{h}+1}\left[1 + \sum_{h=1}^{\infty} 2^{h-2^h}\right] \le 3.563 \times 2^{\tilde{h}} \le 7.126\tilde{\tau} < 8\tilde{\tau}. \end{aligned}$$

To bound the tails of F, for $h \ge 0$ we note by similar estimates that

$$P\left(F > 2^{\tilde{h}+h} - 1\right) = \prod_{g=0}^{\tilde{h}+h-1} \text{sep}_{\hat{0}}(2^g) \le \prod_{g=\tilde{h}}^{\tilde{h}+h-1} \text{sep}_{\hat{0}}(2^g) \le \prod_{g=0}^{h-1} 2^{-2^g} = 2^{-(2^h-1)}. \qquad \square$$

9. The new algorithm and strong stationary times.

When the basic two-phase routine of the algorithm in Section 7.2 is run for an interval of length t, we have already noted at (7.3) and (8.1) that the probability of output is $1 - \text{sep}_{\hat{0}}(t)$. According to a standard theorem ([2]; see also Chapter 9 of [3]), therefore, this probability equals $P(T \le t)$, where T is a *minimal strong stationary time* (MSST, called a *time to stationarity* in [13]) for the $\mathbf{P}$-chain started at $\hat{0}$. We will show that this is no coincidence: the algorithm is intimately connected with the construction of a MSST in just the same way that Algorithm PW is connected with the construction of a backward coupling time. The connections provide further insight into how the algorithms work.

For completeness, we briefly review the notion of strong stationary time SST; for a fuller development, see [1, 2, 3, 13]. In the language of computer science, a *randomized stopping time* for a process $\mathbf{X} = (\mathbf{X}_t)_{t\ge 0}$ is an on-line stopping rule that is allowed to use randomness independent of $\mathbf{X}$.

DEFINITION 9.1. Let $\mathbf{X}$ be an ergodic Markov chain with finite state space and stationary distribution π. A randomized stopping time T is said to be a *strong stationary time* for $\mathbf{X}$ if (1) $\mathbf{X}_T \sim \pi$ and (2) T and $\mathbf{X}_T$ are independent random variables; equivalently, if

$$P(\mathbf{X}_T = \mathbf{x} \mid T = t) = \pi(\mathbf{x}) \quad \text{for all } \mathbf{x}, t. \tag{9.1}$$

As mentioned above, a well-known result [2] implies that for any finite-state ergodic chain (started in any fixed state $\mathbf{x}_0$) there exists a SST T^* that is *minimal* (stochastically smallest), satisfying $\text{sep}_{\mathbf{x}_0}(t) = P(T^* > t) \le P(T > t)$ for any t and any SST T.

9.1. *Construction of a strong stationary time.* Throughout Section 9 we suppose that the monotone case (in the sense of Definitions 4.4 and 4.2) is satisfied by the time-reversed chain $\tilde{\mathbf{P}}$ with monotone transition rule $\tilde{f}$. Consider the construction of the reverse trajectory $\tilde{\mathbf{Y}}$ in the second phase of the routine. Suppose that $\tilde{\mathbf{Y}}_{t-s} = \mathbf{y}$ has been constructed and we wish to choose the value $\mathbf{y}'$ of $\tilde{\mathbf{Y}}_{t-s+1}$. Let $\tilde{\mathbf{X}}_{t-s} = \mathbf{X}_s = \mathbf{x}$ and $\tilde{\mathbf{X}}_{t-s+1} = \mathbf{X}_{s-1} = \mathbf{x}'$ be the corresponding values determined in the first phase. As explained following (7.1), $\mathbf{y}'$ can be chosen by generating a random variable $\hat{\mathbf{U}}_s$ whose distribution is the conditional distribution of $\mathbf{U}$ given $\tilde{f}(\mathbf{x}, \mathbf{U}) = \mathbf{x}'$ and setting $\mathbf{y}' = \tilde{f}(\mathbf{y}, \hat{\mathbf{U}}_s)$.

Notice that the values $\hat{\mathbf{U}}_1, \dots, \hat{\mathbf{U}}_t$ could be generated, one at a time, during the first phase as the trajectory $(\mathbf{X}_1, \dots, \mathbf{X}_t)$ evolves from $\mathbf{X}_0 = \hat{0}$. At each (forward) epoch s we could then ask the following question: which values of $\tilde{\mathbf{Y}}_{t-s}$ [to be determined by the future specification of $(\mathbf{X}_{s+1}, \dots, \mathbf{X}_t)$] would yield $\tilde{\mathbf{Y}}_t = \hat{0}$? Since

$$\tilde{\mathbf{Y}}_t = \tilde{f}(\cdots \tilde{f}(\tilde{f}(\tilde{\mathbf{Y}}_{t-s}, \hat{\mathbf{U}}_s), \hat{\mathbf{U}}_{s-1}) \cdots, \hat{\mathbf{U}}_1),$$

the set of values is precisely

$$\mathbf{X}_s^* := \{\mathbf{y} \in \mathscr{S}: \tilde{f}(\cdots \tilde{f}(\tilde{f}(\mathbf{y}, \hat{\mathbf{U}}_s), \hat{\mathbf{U}}_{s-1}) \cdots, \hat{\mathbf{U}}_1) = \hat{0}\},$$

with $\mathbf{X}_0^* := \{\hat{0}\}$. Several observations follow immmediately.

1. $\mathbf{X}_s \in \mathbf{X}_s^*$; in particular, $\mathbf{X}_s^* \neq \varnothing$.
2. $\mathbf{X}_s^* = \{\mathbf{y} \in \mathscr{S}: \tilde{f}(\mathbf{y}, \hat{\mathbf{U}}_s) \in \mathbf{X}_{s-1}^*\} = \tilde{f}(\cdot, \hat{\mathbf{U}}_s)^{-1}(\mathbf{X}_{s-1}^*)$; in particular, if $\mathbf{X}_{s-1}^* = \mathscr{S}$, then $\mathbf{X}_s^* = \mathscr{S}$.
3. Since $\tilde{f}$ is increasing in its first argument, $\mathbf{X}_s^*$ is an order ideal in $\mathscr{S}$. In particular, $\mathbf{X}_s^* = \mathscr{S}$ if and only if $\hat{1} \in \mathbf{X}_s^*$.
4. The two-phase routine produces output ($= \mathbf{X}_t$) if and only if $\mathbf{X}_t^* = \mathscr{S}$.

Since each $\mathbf{X}_s^*$ is constructed using only $(\mathbf{X}_0, \dots, \mathbf{X}_s)$ and independent randomness, the same construction can be carried out on $\{0, 1, \dots\}$ rather than on the finite time interval $\{0, 1, \dots, t\}$. Suppose this is done, and let

$$T := \inf\{s \geq 0: \mathbf{X}_s^* = \mathscr{S}\}. \tag{9.2}$$

THEOREM 9.2. *T is a minimal strong stationary time for the chain $\mathbf{X}$ started in $\hat{0}$.*

Theorem 9.2 will be proved in the next subsection by applying the theory of strong stationary duality [13].

REMARK 9.3. (a) Theorem 9.2 establishes the validity of an alternative interruptible algorithm for perfect sampling, to wit: construct the process $\mathbf{X}^*$ as described above, stop at the first time T such that $\mathbf{X}_T^* = \mathscr{S}$, and output $\mathbf{X}_T$. Compared with the algorithm in Section 7.2, this algorithm has the advantage of not requiring a second (reverse trajectory) phase.

(b) However, this alternative algorithm is not generally practicable. The problem is that the process $\mathbf{X}^*$ takes values in the space $\mathscr{S}^* = J(\mathscr{S}) \setminus \{\varnothing\}$ of nonempty order ideals of $\mathscr{S}$. Since an order ideal is equivalently represented by its antichain of maximal elements (e.g., [35]), the size (as a subset of $\mathscr{S}$) of an element of $\mathscr{S}^*$ can be as large as the width (maximum size of an antichain) of the poset $\mathscr{S}$. For example, if (as in the case of attractive spin systems) $\mathscr{S}$ is a Boolean algebra, that is, the power set of an n-element set ordered by inclusion, then the width is $\binom{n}{\lfloor n/2 \rfloor} \sim 2^n/\sqrt{\pi n/2}$ (corresponding to the antichain of subsets of size k, where $k = \lfloor n/2 \rfloor$ or $k = \lceil n/2 \rceil$) by a theorem of Sperner [34]. Thus, simply to record a single value $\mathbf{X}_s^*$ would use a prohibitive amount of memory. Additionally, for complex examples such as RSU chains, the computation $\mathbf{X}_s^* = \tilde{f}(\cdot, \hat{\mathbf{U}}_s)^{-1}(\mathbf{X}_{s-1}^*)$ is infeasible.

(c) Consider again the algorithm in (a). According to observations 2 and 3 above, we can test whether $T \le t$ by checking whether

$$\tilde{f}(\cdots \tilde{f}(\tilde{f}(\hat{1}, \hat{\mathbf{U}}_t), \hat{\mathbf{U}}_{t-1}) \cdots, \hat{\mathbf{U}}_1) = \hat{0}.$$

This is precisely the function of the second phase of the routine in the algorithm of Section 7.2! However, note that the total computation time (in number of transitions) in determining the value of T in this way is $\sum_{s=1}^{T} s = \Theta(T^2)$. The “erase, double t, and start over” strategy of the algorithm in Section 7.2 is designed to eliminate the time-squaring slowdown. This strategy is not new: Algorithm PW uses the same technique in conjunction with a backward coupling time T.

(d) The fact that the time T of Theorem 9.2 satisfies the “strong” requirement of independence of T and $\mathbf{X}_T$ in the definition of strong stationary time lies at the heart of why the algorithm of Section 7.2 is interruptible, that is, unbiased with respect to user impatience.

(e) Because T is a *minimal* SST, the algorithm in Section 7.2 has some claim to near-optimality. Indeed, were we only to run the $\mathbf{P}$-chain for t steps from state $\hat{0}$, the highest unconditional probability with which we could output the terminal state as an observation from π is $1 - \mathrm{sep}_{\hat{0}}(t)$. Our routine does just this, at the additional expense only of the reverse trajectory phase.

9.2. *Strong stationary duality.* In this subsection we prove Theorem 9.2 using terminology, notation, and results from [13] for strong stationary duality. Let $\mathbf{X}^*$ be the stochastic process constructed in Section 9.1, and let $\mathscr{S}^*$ denote the collection of nonempty order ideals in $\mathscr{S}$.

THEOREM 9.4. *$\mathbf{X}^*$ is marginally a Markov chain with state space $\mathscr{S}^*$ and is, moreover, a sharp set-valued strong stationary dual for $\mathbf{X}$.*

We offer a brief explanation. “$\mathbf{X}^*$ is a set-valued strong stationary dual” means that $\mathbf{X}$ is linked to $\mathbf{X}^*$ by Λ: $\mathscr{S}^* \times \mathscr{S} \to [0, 1]$, where

$$\Lambda(\mathbf{x}^*, \mathbf{x}) := \begin{cases} \pi(\mathbf{x})/H(\mathbf{x}^*), & \text{if } \mathbf{x} \in \mathbf{x}^*, \\ 0, & \text{otherwise} \end{cases} \tag{9.3}$$

is the link of truncated stationary distributions, in the sense that

$$\mathscr{L}(\mathbf{X}_t \mid \mathbf{X}_0^* = \mathbf{x}_0^*, \dots, \mathbf{X}_t^* = \mathbf{x}_t^*) = \Lambda(\mathbf{x}_t^*, \cdot). \tag{9.4}$$

In (9.3), we have written H for the probability measure corresponding to the mass function π:

$$H(\mathbf{x}^*) = \sum_{\mathbf{x}\in\mathbf{x}^*} \pi(\mathbf{x}).$$

Equation (9.4) (with $\mathbf{x}_t^* = \mathscr{S}$) immediately implies that T of (9.2) has the defining property (9.1) of a strong stationary time. The assertion that $\mathbf{X}^*$ is a "sharp" dual is precisely that the SST T is minimal. Thus Theorem 9.2 is a corollary of Theorem 9.4, and (9.4) additionally gives distributional information about the chain $\mathbf{X}$ prior to stopping.

The remainder of this subsection is rather technical and makes frequent reference to [13]; the reader may wish to skip it upon first reading. The next two lemmas will be used to prove Theorem 9.4.

LEMMA 9.5. *Let the probability measures π_0 on $\mathscr{S}$ and π_0^* on $\mathscr{S}^*$ be unit masses at $\hat{0}$ and at $\{\hat{0}\}$, respectively, and define*

$$\mathbf{P}^*(\mathbf{x}^*, \mathbf{y}^*) := \frac{H(\mathbf{y}^*)}{H(\mathbf{x}^*)} P(\tilde{f}(\cdot, \mathbf{U})^{-1}(\mathbf{x}^*) = \mathbf{y}^*), \qquad \mathbf{x}^* \in \mathscr{S}^*,\ \mathbf{y}^* \in \mathscr{S}^*.$$

Then $\mathbf{P}^*$ is a transition matrix on $\mathscr{S}^*$, and $(\pi_0^*, \mathbf{P}^*)$ is algebraically dual to $(\pi_0, \mathbf{P})$ with respect to the link Λ, in the sense that

$$\text{(i) } \pi_0 = \pi_0^*\Lambda \quad \textit{and} \quad \text{(ii) } \Lambda\mathbf{P} = \mathbf{P}^*\Lambda.$$

PROOF. Part (i) is clear, and from part (ii) it follows that $\sum_{\mathbf{y}^*} \mathbf{P}^*(\mathbf{x}^*, \mathbf{y}^*) = 1$ for each $\mathbf{x}^*$. So the following calculation proving (ii) suffices:

$$\begin{aligned}
(\Lambda\mathbf{P})(\mathbf{x}^*, \mathbf{y}) &= \frac{1}{H(\mathbf{x}^*)} \sum_{\mathbf{x}\in\mathbf{x}^*} \pi(\mathbf{x})\mathbf{P}(\mathbf{x}, \mathbf{y}) = \frac{\pi(\mathbf{y})}{H(\mathbf{x}^*)} \sum_{\mathbf{x}\in\mathbf{x}^*} \tilde{\mathbf{P}}(\mathbf{y}, \mathbf{x}) \\
&= \frac{\pi(\mathbf{y})}{H(\mathbf{x}^*)} P(\tilde{f}(\mathbf{y}, \mathbf{U}) \in \mathbf{x}^*) = \frac{\pi(\mathbf{y})}{H(\mathbf{x}^*)} P(\tilde{f}(\cdot, \mathbf{U})^{-1}(\mathbf{x}^*) \ni \mathbf{y}) \\
&= \frac{\pi(\mathbf{y})}{H(\mathbf{x}^*)} \sum_{\mathbf{y}^*\ni\mathbf{y}} P(\tilde{f}(\cdot, \mathbf{U})^{-1}(\mathbf{x}^*) = \mathbf{y}^*) = \frac{\pi(\mathbf{y})}{H(\mathbf{x}^*)} \sum_{\mathbf{y}^*\ni\mathbf{y}} \frac{H(\mathbf{x}^*)}{H(\mathbf{y}^*)} \mathbf{P}^*(\mathbf{x}^*, \mathbf{y}^*) \\
&= \sum_{\mathbf{y}^*} \mathbf{P}^*(\mathbf{x}^*, \mathbf{y}^*)\Lambda(\mathbf{y}^*, \mathbf{y}) = (\mathbf{P}^*\Lambda)(\mathbf{x}^*, \mathbf{y}). \qquad \square
\end{aligned}$$

REMARK 9.6. (a) Lemma 9.5 is a partial extension of Theorem 4.6 in [13] from linearly ordered to partially ordered state spaces.

(b) As will be explained in a future paper [18], $\mathbf{P}^*$ is the H-transform of a (generally nonunique) Siegmund dual of the time-reversal of $\mathbf{P}$, extending Theorem 5.5 in [13].

In the next lemma, $\boldsymbol{\sigma}_0$ and $\mathbf{Q}$ play roles analogous to those of $\boldsymbol{\pi}_0$ and $\mathbf{P}$ on the left sides of (2.20) and (2.21), respectively, in [13]. As the proof will make clear, there is in the present setting an important deviation from the construction of (2.21) in [13]. The conclusion that $\mathbf{X}^*$ and $\mathbf{X}$ are marginally Markov is noteworthy, since functions of Markov chains are not generally Markov.

LEMMA 9.7. *On the bivariate state space* $\{(\mathbf{x}^*, \mathbf{x}) \in \mathscr{S}^* \times \mathscr{S} : \mathbf{x} \in \mathbf{x}^*\}$, *let* $\boldsymbol{\sigma}_0$ *denote unit mass at* $(\{\hat{0}\}, \hat{0})$ *and* $\mathbf{Q}$ *the transition matrix*

$$\mathbf{Q}((\mathbf{x}^*, \mathbf{x}), (\mathbf{y}^*, \mathbf{y})) := \frac{\pi(\mathbf{y})}{\pi(\mathbf{x})} P(\tilde{f}(\mathbf{y}, \mathbf{U}) = \mathbf{x},\ \tilde{f}(\cdot, \mathbf{U})^{-1}(\mathbf{x}^*) = \mathbf{y}^*).$$

If $(\mathbf{X}_t^*, \mathbf{X}_t)_{t \ge 0}$ *is a bivariate Markov chain with initial distribution* $\boldsymbol{\sigma}_0$ *and transition matrix* $\mathbf{Q}$, *then* $(\mathbf{X}_t^*)_{t\ge 0}$ *and* $(\mathbf{X}_t)_{t \ge 0}$ *are marginally Markov chains, with respective initial distributions* π_0^* *and* π_0 *and respective transition matrices* $\mathbf{P}^*$ *and* $\mathbf{P}$. *Further, equations* (2.13)–(2.16) *and* (2.28) *of* [13] *hold in the present setting, and* (2.27) *there holds here in the modified form*

$$\tag{9.5} \begin{aligned} &P\big(\mathbf{X}_t^* = \mathbf{x}_t^* \mid (\mathbf{X}_0^*, \mathbf{X}_0) = (\mathbf{x}_0^*, \mathbf{x}_0); \ldots; (\mathbf{X}_{t-1}^*, \mathbf{X}_{t-1}) = (\mathbf{x}_{t-1}^*, \mathbf{x}_{t-1});\ \mathbf{X}_t = \mathbf{x}_t\big) \\ &\qquad = \frac{\mathbf{Q}((\mathbf{x}_{t-1}^*, \mathbf{x}_{t-1}), (\mathbf{x}_t^*, \mathbf{x}_t))}{\mathbf{P}(\mathbf{x}_{t-1}, \mathbf{x}_t)}. \end{aligned}$$

PROOF. As at (2.22) in [13], define $\Delta = \mathbf{P}^*\Lambda = \Lambda\mathbf{P}$; that is,

$$\Delta(\mathbf{x}^*, \mathbf{y}) := \sum_{\mathbf{y}^*} \mathbf{P}^*(\mathbf{x}^*, \mathbf{y}^*)\Lambda(\mathbf{y}^*, \mathbf{y}) = \sum_{\mathbf{x}} \Lambda(\mathbf{x}^*, \mathbf{x})\mathbf{P}(\mathbf{x}, \mathbf{y}).$$

We apply Remark 2.23(c) in [13] and, for the assertions (2.13) and (2.28) in [13] and (9.5) here, an extension of Remark 2.23(b) in [13] that is fairly routine to verify. According to these remarks, which take as granted the *algebraic duality* equations $\pi_0 = \pi_0^*\Lambda$ and $\Lambda\mathbf{P} = \mathbf{P}^*\Lambda$ established here in Lemma 9.5, it is enough to show for fixed $\mathbf{x}^*$ and $\mathbf{y}$ that

$$\tag{9.6} \mathbf{p}(\mathbf{x}, \mathbf{y}^* \mid \mathbf{x}^*, \mathbf{y}) := \frac{\Lambda(\mathbf{x}^*, \mathbf{x})}{\Delta(\mathbf{x}^*, \mathbf{y})} \mathbf{Q}((\mathbf{x}^*, \mathbf{x}), (\mathbf{y}^*, \mathbf{y})), \qquad \mathbf{x} \in \mathbf{x}^*,\ \mathbf{y}^* \ni \mathbf{y}$$

defines a probability mass function in $\mathbf{x}$ and $\mathbf{y}^*$ with respective marginals

$$\tag{9.7} \Lambda(\mathbf{x}^*, \mathbf{x})\mathbf{P}(\mathbf{x}, \mathbf{y})/\Delta(\mathbf{x}^*, \mathbf{y}) \quad \text{and} \quad \mathbf{P}^*(\mathbf{x}^*, \mathbf{y}^*)\Lambda(\mathbf{y}^*, \mathbf{y})/\Delta(\mathbf{x}^*, \mathbf{y}).$$

Indeed, for $\mathbf{x} \in \mathbf{x}^*$, the calculation

$$\begin{aligned} \sum_{\mathbf{y}^* \in \mathbf{y}} \mathbf{Q}((\mathbf{x}^*, \mathbf{x}), (\mathbf{y}^*, \mathbf{y})) &= \frac{\pi(\mathbf{y})}{\pi(\mathbf{x})} P(\tilde{f}(\mathbf{y}, \mathbf{U}) = \mathbf{x},\ \tilde{f}(\cdot, \mathbf{U})^{-1}(\mathbf{x}^*) \ni \mathbf{y}) \\ &= \frac{\pi(\mathbf{y})}{\pi(\mathbf{x})} P(\tilde{f}(\mathbf{y}, \mathbf{U}) = \mathbf{x}) = \frac{\pi(\mathbf{y})}{\pi(\mathbf{x})} \tilde{\mathbf{P}}(\mathbf{y}, \mathbf{x}) = \mathbf{P}(\mathbf{x}, \mathbf{y}) \end{aligned}$$

verifies the first marginal in (9.7), and for $\mathbf{y}^* \ni \mathbf{y}$ the calculation

$$\begin{aligned}
&\sum_{\mathbf{x}\in\mathbf{x}^*} \Lambda(\mathbf{x}^*, \mathbf{x})\mathbf{Q}((\mathbf{x}^*, \mathbf{x}), (\mathbf{y}^*, \mathbf{y}))\\
&\quad = \sum_{\mathbf{x}\in\mathbf{x}^*} \frac{\pi(\mathbf{x})}{H(\mathbf{x}^*)} \frac{\pi(\mathbf{y})}{\pi(\mathbf{x})} P(\tilde{f}(\mathbf{y}, \mathbf{U}) = \mathbf{x}, \tilde{f}(\cdot, \mathbf{U})^{-1}(\mathbf{x}^*) = \mathbf{y}^*)\\
&\quad = \frac{\pi(\mathbf{y})}{H(\mathbf{x}^*)} P(\tilde{f}(\mathbf{y}, \mathbf{U}) \in \mathbf{x}^*, \tilde{f}(\cdot, \mathbf{U})^{-1}(\mathbf{x}^*) = \mathbf{y}^*)\\
&\quad = \frac{\pi(\mathbf{y})}{H(\mathbf{x}^*)} P(\tilde{f}(\cdot, \mathbf{U})^{-1}(\mathbf{x}^*) = \mathbf{y}^* \ni \mathbf{y})\\
&\quad = \frac{\pi(\mathbf{y})}{H(\mathbf{x}^*)} P(\tilde{f}(\cdot, \mathbf{U})^{-1}(\mathbf{x}^*) = \mathbf{y}^*)\\
&\quad = \mathbf{P}^*(\mathbf{x}^*, \mathbf{y}^*)\Lambda(\mathbf{y}^*, \mathbf{y})
\end{aligned}$$

verifies the second marginal. □

REMARK 9.8. The substantial new wrinkle in our present construction of the bivariate $\mathbf{Q}$ is that now (9.6) is used in place of the product of the marginal distributions (9.7) in constructing $\mathbf{Q}((\mathbf{x}^*, \mathbf{x}), (\mathbf{y}^*, \mathbf{y}))$ as

$$\Delta(\mathbf{x}^*, \mathbf{y})\mathbf{p}(\mathbf{x}, \mathbf{y}^* \mid \mathbf{x}^*, \mathbf{y})/\Lambda(\mathbf{x}^*, \mathbf{x}).$$

PROOF OF THEOREM 9.4. Sharpness will follow from Remark 2.39 in [13], because the only state in $\mathscr{S}^*$ containing $\hat{1} \in \mathscr{S}$ is the absorbing state $\mathscr{S}$. So we need only verify that $\mathbf{X}^*$ as constructed in Section 9.1 is a strong stationary dual (SSD) Markov chain for $\mathbf{X}$. Following the same arguments as in Section 2.4 of [13], as $\mathbf{X}$ evolves we can build a SSD Markov chain $\mathbf{X}^*$ so that the resulting bivariate process is a $(\boldsymbol{\sigma}_0, \mathbf{Q})$ Markov chain as in Lemma 9.7, as follows. Suppose that $\mathbf{X}_0 = \mathbf{x}_0 = \hat{0}$, $\mathbf{X}_1 = \mathbf{x}_1, \dots, \mathbf{X}_{t-1} = \mathbf{x}_{t-1}$ have been observed and that $\mathbf{X}_0^* = \mathbf{x}_0^* = \{\hat{0}\}$, $\mathbf{X}_1^* = \mathbf{x}_1^*, \dots, \mathbf{X}_{t-1}^* = \mathbf{x}_{t-1}^*$ have been chosen. When $\mathbf{X}_t = \mathbf{x}_t$ is observed, set $\mathbf{X}_t^* = \mathbf{x}_t^*$ with (conditional) probability

$$\begin{aligned}
\frac{\mathbf{Q}((\mathbf{x}_{t-1}^*, \mathbf{x}_{t-1}), (\mathbf{x}_t^*, \mathbf{x}_t))}{\mathbf{P}(\mathbf{x}_{t-1}, \mathbf{x}_t)} &= \frac{P(f(\mathbf{x}_t, \mathbf{U}) = \mathbf{x}_{t-1}, f(\cdot, \mathbf{U})^{-1}(\mathbf{x}_{t-1}^*) = \mathbf{x}_t^*)}{\tilde{\mathbf{P}}(\mathbf{x}_t, \mathbf{x}_{t-1})}\\
&= P(f(\cdot, \mathbf{U})^{-1}(\mathbf{x}_{t-1}^*) = \mathbf{x}_t^* \mid f(\mathbf{x}_t, \mathbf{U}) = \mathbf{x}_{t-1}).
\end{aligned}$$

But this is exactly the construction in Section 9.1, so Theorem 9.4 is proved. □

REMARK 9.9. (a) Reinforcing Remark 9.8, for our SSD construction the value assigned to $\mathbf{X}_t^*$ depends on $\mathbf{X}_{t-1}^*$, $\mathbf{X}_{t-1}$, $\mathbf{X}_t$ and independent randomness, while the corresponding construction in Section 2.4 of [13] does not use $\mathbf{X}_{t-1}$.

(b) In addition to providing a method for stationary sampling, duality has converted a convergence-to-stationarity problem for $\mathbf{X}$ into an absorption-time problem for the chain $\mathbf{X}^*$. Unfortunately, for complex examples such as

RSU chains, the latter problem seems no more analytically tractable than the former.

10. A fine-tuned algorithm for RSU chains.

10.1. *How to handle a hold.* We return to the setting of Example 7.2 in order to complete the specialization of the algorithm of Section 7.2 to RSU chains. The question left open was the following. Suppose that a hold, say $\tilde{\mathbf{X}}_{t-s}(=\mathbf{X}_s) = \tilde{\mathbf{X}}_{t-s+1}(=\mathbf{X}_{s-1}) = \mathbf{x}$, has been observed and that the value $\tilde{\mathbf{Y}}_{t-s} = \mathbf{y}$ has been assigned. How does one make the assignment of value to $\tilde{\mathbf{Y}}_{t-s+1}$? As explained following (7.1), one chooses the value $\mathbf{y}'$ with probability $P(f(\mathbf{y}, \mathbf{U}) = \mathbf{y}' \mid f(\mathbf{x}, \mathbf{U}) = \mathbf{x})$, where f is the monotone transition rule of (4.2). [In this example $\mathbf{P} = \tilde{\mathbf{P}}$, so f also serves as the $\tilde{f}$ in (7.1).] The rather involved calculation of the conditional distribution of $\mathbf{U}$ given $f(\mathbf{x}, \mathbf{U}) = \mathbf{x}$ can be avoided altogether by making efficient use of randomness. We assume that forward transitions for $\mathbf{X}$ are also generated using the transition rule f. Then the forward hold was produced in the first place by generating $\mathbf{U} = (U_1, U_2)$ uniformly distributed on $\mathscr{U} = \{1, \ldots, n\} \times [0, 1]$ and finding $f(\mathbf{x}, \mathbf{U}) = \mathbf{x}$. So one can simply store and later reuse this value to set $\tilde{\mathbf{Y}}_{t-s+1} = f(\tilde{\mathbf{Y}}_{t-s}, \mathbf{U})$.

REMARK 10.1. If desired, one can use the same idea to save on random number generation for nonholds. Recall Example 7.2 and consider an $\tilde{\mathbf{X}}$-transition from $\mathbf{x}$ to $\mathbf{x}'$, that is, an $\mathbf{X}$-transition from $\mathbf{x}'$ to $\mathbf{x}$, where $v = v_u$ and $x_v = +1$ and $\mathbf{x}' = (\mathbf{x};\ x_v \leftarrow -1)$. Observe

$$\begin{aligned}
&\mathscr{L}((U_1, U_2) \mid f(\mathbf{x}', U_1, U_2) = \mathbf{x}) \\
&\quad = \mathscr{L}((U_1, U_2) \mid U_1 = u, f_v(\mathbf{x}', U_2) = \mathbf{x}) = \mathscr{L}((u, U_2) \mid U_2 > \mathbf{P}_v(\mathbf{x}', \mathbf{x}')) \\
&\quad = \mathscr{L}\left((u, U_2) \,\middle|\, U_2 > \frac{\pi(\mathbf{x}')}{\pi(\mathbf{x}) + \pi(\mathbf{x}')}\right) = \mathscr{L}\left(u, 1 - \frac{\pi(\mathbf{x})}{\pi(\mathbf{x}) + \pi(\mathbf{x}')} U_2\right),
\end{aligned}$$

while similarly

$$\begin{aligned}
&\mathscr{L}((U_1, U_2) \mid f(\mathbf{x}, U_1, U_2) = \mathbf{x}') \\
&\quad = \mathscr{L}\left(u, \frac{\pi(\mathbf{x}')}{\pi(\mathbf{x}) + \pi(\mathbf{x}')} U_2\right) \\
&\quad = \mathscr{L}\left(\left(U_1, \frac{\pi(\mathbf{x}')}{\pi(\mathbf{x})}(1 - U_2)\right) \mid f(\mathbf{x}', U_1, U_2) = \mathbf{x}\right).
\end{aligned}$$

So if $\mathbf{X}_s = f(\mathbf{X}_{s-1}, U_1, U_2) = (\mathbf{X}_{s-1};\ (\mathbf{X}_{s-1})_{v_{U_1}} \leftarrow +1) \neq \mathbf{X}_{s-1}$, then we can construct $\tilde{\mathbf{Y}}_{t-s+1}$ via

$$\tilde{\mathbf{Y}}_{t-s+1} = f\left(\tilde{\mathbf{Y}}_{t-s}, U_1, \frac{\pi(\mathbf{X}_{s-1})}{\pi(\mathbf{X}_s)}(1 - U_2)\right).$$

10.2. *An algorithm requiring less memory: Algorithm RSUS.* As detailed in Section 8 (see especially Theorem 8.1), the algorithm of Section 7.2, which stores the entire forward trajectory, runs in time of the order of the mixing time $\tau_1^{(1)}$ of $\mathbf{P}$ and space of order $\tau_1^{(1)}\sigma$ where σ is the memory required to store a single state. In the case of RSU chains, $\sigma = n$ and so the space requirement is of order $\tau_1^{(1)}n$. We cannot hope to reduce the order of the running time. However, our algorithm's space requirement is typically much larger than the corresponding bound $O(\tau_1(\log n)^2 + n)$ [of which only $O(n)$ need be read-write memory] for PW.

Further, recall from Section 6.2 that Algorithm PW, when implemented with a seeded pseudorandom number generator, uses space of order n, where we may assume that the mixing time for $\mathbf{P}$ grows at most polynomially in n. Even at this diminished space requirement, Wilson [38] has noted that space, not time, is the limiting resource for PW in certain interesting applications. Thus we seek to modify our algorithm for RSU chains to reduce the memory requirement.

Suppose first that truly random bits are used. Then the space required can be reduced to $O(\tau_1^{(1)} \log n + n)$ by an imputation trick like that described for PW at the end of Section 6.2. Again it suffices to store the U_1-values and ternary digits rather than U_2-values. As the first phase evolves, one stores for each step one of the three digit values if the step spins site v_{U_1} from $+1$ to -1, a second value if it spins site v_{U_1} from -1 to $+1$, and the third value if the step is a hold. These digits can then be used for two purposes during the second phase: (1) one can redetermine $\tilde{\mathbf{X}}$, one step at a time; and (2) it is a simple matter to impute U_2-values to be used for the synchronistic construction of $\tilde{\mathbf{Y}}$. Note that the hold case is no more difficult than the other cases here, since one can condition on the stored value of U_1. We leave the details of the imputation to the reader as an exercise.

When our algorithm is implemented using a seeded pseudorandom number generator, as we assume for the remainder of Section 10, one can even reduce the space needed to order n, by eliminating the need to store *any* information about the $\mathbf{U}$-values, but at the expense of a great deal of time. When one needs to know the value of $\tilde{\mathbf{X}}_s$, that is, of $\mathbf{X}_{t-s}$, one simply recomputes the entire initial trajectory segment $(\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_{t-s})$ by regenerating the appropriate $\mathbf{U}$-values. However, this squares the order of the algorithm's run time, and so is unsuitable.

Comments by Cowen [11] inspired the following compromise, which we christen Algorithm RSUS (for Random Site Updating Sampler), and which is based on the observation for the basic algorithm that one can reconstruct a trajectory segment $(\tilde{\mathbf{X}}_r, \ldots, \tilde{\mathbf{X}}_s) = (\mathbf{X}_{t-s}, \ldots, \mathbf{X}_{t-r})$ and build the corresponding segment of $\tilde{\mathbf{Y}}$ from knowledge of just $\mathbf{X}_{t-s}$, the corresponding generator seed at that time, and $\tilde{\mathbf{Y}}_r$.

Algorithm RSUS differs from the basic algorithm only in the way that the ith iteration (with $t = 2^{i-1}$) is carried out ($i = 1, 2, \ldots$). For Algorithm RSUS, this is done by calling

$$\text{CoupledReverse}(i-1, \hat{0}, \text{initialseed}, \hat{1}).$$

The procedure CoupledReverse$(h, \mathbf{x}_0, \text{startseed}, \mathbf{y})$, listed below, takes as input the binary logarithm h of an interval length 2^h, an initial configuration

$\mathbf{x}_0$ and uniform random number generator seed startseed for a forward trajectory ($\mathbf{X}$) and an initial configuration $\mathbf{y}$ for an appropriately coupled reverse trajectory ($\tilde{\mathbf{Y}}$). It returns the final configuration in the $\tilde{\mathbf{Y}}$-trajectory. The procedure ForwardStep($\mathbf{x}$, seed) is assumed to take an initial configuration $\mathbf{x}$ and a given seed; use first the generator to produce a $\mathbf{U}$-value and then the transition rule f of (4.2), and return the pair ($f(\mathbf{x}, \mathbf{U})$, correspondingly updated seed). The procedure CoupledReverseStep($\mathbf{x}_0$, startseed, $\mathbf{y}$) is assumed to carry out CoupledReverse($0, \mathbf{x}_0$, startseed, $\mathbf{y}$) correctly; we do not present the details of CoupledReverseStep, but note for future reference that it involves a single call to the procedure ForwardStep.

```
CoupledReverse($h, \mathbf{x}_0$, startseed, $\mathbf{y}$):

if $h = 0$
    return CoupledReverseStep($\mathbf{x}_0$, startseed, $\mathbf{y}$)
else
    $\mathbf{x} \leftarrow \mathbf{x}_0$
    seed $\leftarrow$ startseed
    for $g \leftarrow 1$ to $2^{h-1}$
        ($\mathbf{x}$, seed) $\leftarrow$ ForwardStep($\mathbf{x}$, seed)
    $\mathbf{y} \leftarrow$ CoupledReverse($h-1, \mathbf{x}$, seed, $\mathbf{y}$)
    return CoupledReverse($h-1, \mathbf{x}_0$, startseed, $\mathbf{y}$)
```

10.3. *Performance of Algorithm RSUS.* The following theorem briefly summarizes the time and space requirements of Algorithm RSUS.

THEOREM 10.2. *The expected running time of Algorithm RSUS is $O(\tau_1^{(1)} \log \tau_1^{(1)})$, and the expected space requirement for dynamic memory is $O(n \log \tau_1^{(1)})$. Here $\tau_1^{(1)}$ is the mixing time parameter given by* (8.5) *and* (8.4) *for the random site update chain $\mathbf{P} = n^{-1} \sum_v \mathbf{P}_v$.*

PROOF. From the description of Algorithm RSUS it is clear that the run time is linear in the number of calls, say F^*, to the procedure ForwardStep. The number of calls to ForwardStep used in CoupledReverse($h, \mathbf{x}_0$, startseed, $\mathbf{y}$), say f_h, is deterministically given by the recurrence relation

$$f_h = 2^{h-1} + 2f_{h-1} \qquad \text{for } h \geq 1, \text{ with } f_0 = 1,$$

which is easily solved to yield

$$f_h = \left(\frac{h}{2} + 1\right) 2^h, \qquad h \geq 0. \tag{10.1}$$

Thus F^* is a random variable taking value $\sum_{h=0}^{i-1} f_h$ with probability $P(I = i)$, $i \geq 1$, where I (with the same value as for the algorithm in Section 7.2) is the number of iterations required. It follows that the analogues of (8.2) and (8.3) for Algorithm RSUS are

$$EF^* = \sum_{i=1}^{\infty} P(I=i) \sum_{h=0}^{i-1} f_h = \sum_{h=0}^{\infty} f_h P(I > h) = \sum_{h=0}^{\infty} f_h \prod_{g=0}^{h-1} \mathrm{sep}_{\hat{0}}(2^g), \tag{10.2}$$

$$P\biggl(F^* > \sum_{g=0}^{h-1} f_g\biggr) = P(I > h) = \prod_{g=0}^{h-1} \mathrm{sep}_{\hat{0}}(2^g), \qquad h \geq 0. \tag{10.3}$$

Proceeding as in the proof of Proposition 8.2 and using the explicit expression (10.1), we find

$$\begin{aligned} EF^* &\leq \sum_{h=0}^{\tilde{h}} f_h + \sum_{h=1}^{\infty} f_{\tilde{h}+h} 2^{-(2^h-1)} \\ &\leq 2^{\tilde{h}}(\tilde{h}+2) + 2^{\tilde{h}}\Biggl[\tilde{h} \sum_{h=1}^{\infty} 2^{h-2^h} + \sum_{h=1}^{\infty}(h+2)2^{h-2^h}\Biggr] \\ &\leq 2^{\tilde{h}}(2\tilde{h}+5) = O(\tau_1^{(1)} \log \tau_1^{(1)}). \end{aligned}$$

Also, as in Proposition 8.2, the right tail of F^* is at most geometrically thick.

It is also not hard to see that the (deterministic) memory requirement for a call to CoupledReverse$(h, \mathbf{x}_0, \text{startseed}, \mathbf{y})$ with $h \geq 1$ is linear in nh; the key here is that the memory used for the first recursive call with h reduced to $h-1$ can be reused for the second such call. Since memory can also be erased after each iteration, the memory required is linear in In. But another calculation like that for EF^* gives

$$EI \leq \tilde{h} + 1 + 2 \sum_{h=1}^{\infty} 2^{-2^h} \leq \tilde{h} + 3 = O(\log \tau_1^{(1)}). \qquad \square$$

11. Comparison of algorithms.

11.1. *Mixing times, reversibility, and other updating schemes.* The Propp–Wilson algorithm [31] uses monotone $\mathbf{P}$ while the algorithm introduced in this paper assumes monotone time reversal $\tilde{\mathbf{P}}$. In comparing the algorithms, then, it makes sense to assume that the monotone t.m. $\mathbf{M}$ taken as $\mathbf{P}$ in Algorithm PW is taken as $\tilde{\mathbf{P}}$ in the new algorithm, and we shall do so without further comment.

Even without assuming any monotonicity, it is clear from the definition (8.4) of $\mathrm{sep}(t)$ that its value does not change under time reversal. In order to compare the performance of the algorithms, therefore, we need to compare the variation threshold τ_1 of (6.2) with the separation threshold $\tau_1^{(1)}$ of (8.5). This is easy in the case of reversible chains (e.g., [3], Chapter 4).

LEMMA 11.1. *For any finite-state ergodic reversible Markov chain,*

$$\tau_1 \leq \tau_1^{(1)} \leq 4\tau_1.$$

The first inequality holds also for nonreversible chains, but there is no universal constant bound on $\tau_1^{(1)}/\tau_1$, even for monotone chains, as demonstrated by Example 6.4. For the monotone t.m. $\mathbf{M} := \mathbf{P}_{\text{mix}}$ there, we have $\tau_1 = 1$ while $\tau_1^{(1)} \geq d$. Moreover, we have the stronger statements that the coalescence time for $\mathbf{M}$ equals 1 with probability at least $1 - 1/d$, while $\text{sep}_{\hat{0}}(t) = 1$ for $\tilde{\mathbf{M}}$ for $t < d$. Thus the algorithm of Section 7.2 with high probability requires at least d times as many forward transitions as does Algorithm PW in this case. Of course, this increase in running time is necessary to correct the sort of large relative errors described in Example 6.4.

It is reasonable to focus attention on reversible chains, since two of the most common methods for simulating a Markov chain with a desired stationary distribution π are the Metropolis algorithm and the Gibbs sampler with random site updating, both of which give reversibility. It is even more common in practice to use the Gibbs sampler $\mathbf{P}_{\text{SSU}} = \mathbf{P}_{v_1}\mathbf{P}_{v_2}\cdots\mathbf{P}_{v_n}$ with *systematic site updating*, where $v_1, \ldots, v_n$ is some fixed ordering of the sites. Since $\tilde{\mathbf{P}}_{\text{SSU}} = \mathbf{P}_{v_n}\mathbf{P}_{v_{n-1}}\cdots\mathbf{P}_{v_1}$, reversibility is lost, but (1) it is easy to implement and, in terms of $\tau_1^{(1)}$, to analyze fine-tuned versions of our user-impatience unbiased algorithm for this and other alternative updating schemes; and (2) absent analytical information about the respective mixing time values, there seems to be no particular reason to prefer $\mathbf{P}_{\text{SSU}}$ over either its multiplicative $(\mathbf{P}_{\text{SSU}}\tilde{\mathbf{P}}_{\text{SSU}})$ or additive $[\frac{1}{2}(\mathbf{P}_{\text{SSU}} + \tilde{\mathbf{P}}_{\text{SSU}})]$ reversiblization.

For balance, we develop a nontrivial nonreversible example in a companion paper [16].

11.2. *Comparison of Propp–Wilson and the new algorithm.* Let $\mathscr{S}$ be a partially ordered state space possessing a unique minimum (respectively, maximum) element $\hat{0}$ (resp., $\hat{1}$). Let $\mathbf{M}$ be a monotone t.m. on $\mathscr{S}$ to be used as $\mathbf{P}$ for the Algorithm PW of Section 5.2 and as $\tilde{\mathbf{P}}$ for our algorithm in Section 7.2. Let τ_1 and $\tau_1^{(1)}$ be the variation and separation thresholds for $\mathbf{M}$ given by (6.2) and (8.5), respectively. We now give a point-by-point comparison of the two algorithms.

Underlying ideas. Backward coupling for PW; acceptance/rejection, or strong stationary times and duality, for ours.

Applicability. PW requires the existence of a monotone transition rule (recall Definition 4.2) for $\mathbf{M}$ (and this is a somewhat stronger requirement than monotonicity of $\mathbf{M}$; recall Remark 4.5); ours does not, but requires the ability to generate transitions from each $\mathbf{K}_{\mathbf{x},\mathbf{y}}(\mathbf{x}', \cdot)$, where (see Section 7.1) $\mathbf{K}_{\mathbf{x},\mathbf{y}}$ (for $\mathbf{x} \leq \mathbf{y}$) is an upward kernel for $\mathbf{M}(\mathbf{x}, \cdot)$ and $\mathbf{M}(\mathbf{y}, \cdot)$. Ours additionally requires the ability to generate transitions from each $\tilde{\mathbf{M}}(\mathbf{x}, \cdot)$; in the nonreversible case, PW does not.

Generalizability. Ideas initiated by Kendall [26] lead to a modification of PW for "anti-monotone" chains, including repulsive spin systems; see Häggström and Nelander [22] for definitions and further development. As for monotone chains, the algorithm maintains only two states. Our algorithm can be similarly modified.

The PW algorithm also possesses the noteworthy feature of allowing for “omnithermal” sampling of attractive spin systems, in effect simultaneously generating one observation for every possible temperature. See Section 3.1 and the remarkable Figure 2 in [31]. Our algorithm similarly can allow for omnithermal sampling.

There exist variants of both algorithms for nonmonotone chains, although neither is very useful for generic chains with large state space, in view of the comment at the outset of Section 4. See [31] for PW; we plan to discuss the variant of our algorithm in future work. There are other schemes to handle general chains: see [37], [29], [4].

Both algorithms are also easily generalized to allow time-inhomogeneous chains, although the performance analyses are greatly complicated in doing so.

Performance. Unlike PW, our algorithm is not subject to user-impatience bias; that is a major theme of the present paper. The expected number of Markov transitions for PW is $O(\tau_1 \log l)$, where l is the length of the longest chain in $\mathscr{S}$; the corresponding guarantee for ours is $O(\tau_1^{(1)})$. If $\mathbf{M}$ is reversible, our bound is smaller; otherwise, our running time can be worse: see Section 11.1. Further, a count of forward transitions does not tell the full running time story. In [16] we present a nonreversible example for which transitions from $\tilde{\mathbf{M}}$ (required for our algorithm but not for PW) take time of larger order of magnitude than those from $\mathbf{M}$. Similarly, generating from the upward kernels might take longer than from $\mathbf{M}$.

From now on we restrict attention to attractive spin systems on n sites. We will consider only the performance of RSU chains, as in Definition 3.1; similar analysis is possible for other updating schemes. We write τ indifferently for τ_1 or $\tau_1^{(1)}$. See Sections 6.2, 10.2, and 10.3 for supporting details.

1. Suppose first that truly random bits are used.
Expected run time. Our guarantee is smaller, $O(\tau)$ vs. $O(\tau \log n)$ for PW.
Expected space. For read/write memory, the guarantee is $O(n+\log \tau)$ for PW and $O(\tau \log n+n)$ for our algorithm. PW requires an additional $O(\tau(\log n)^2)$ bits of read-only memory, plus enough additional read-only memory to store $O(\tau \log n)$ numbers uniformly distributed on [0, 1].

In fairness, we emphasize that the above summary concerns only *bounds*. At least for some examples (e.g., see [16]), the general bound (6.3) for PW can be sharpened to $ET = O(\tau)$. If this sharpened bound holds for a RSU chain, then the expected run time bounds for the two algorithms become equal in magnitude, and the read-only memory bounds for PW are also reduced by a factor of $\log n$.

2. Now suppose that a seeded pseudorandom number generator, as described at the end of Section 6.2, is used. From a probabilistic viewpoint, our analysis treats the numbers generated as if truly random. As discussed at the end of Section 6.2, we may assume that $\tau = O(n^{\alpha})$ for some $\alpha < \infty$.
Expected run time. Our guarantee is $O(\tau \log \tau)$, of same order as PW's $O(\tau \log n)$.

Expected space. Our guarantee is $O(n \log \tau) = O(n \log n)$, which is logarithmically larger than PW's $O(n + \log \tau) = O(n)$.

Acknowledgments. The author thanks David Aldous, Lenore Cowen, Keith Crank, Motoya Machida, Jim Propp and David Wilson for helpful discussions and also thanks anonymous referees for their suggestions (including a simplified proof of Lemma 7.4) which improved the exposition in this paper.

Notes added in proof. Morten Fismen of the Norwegian University of Science and Technology has implemented the author's interruptible perfect sampling algorithm and tested it on the Ising model (personal communication). Elke Thönnes (Perfect simulation of some point processes for the impatient user, Technical Report 317, Dept. Statistics, Univ. Warwick, 1997) extends the algorithm so as to produce perfect samples for the penetrable sphere mixture process and related models.

REFERENCES

[1] ALDOUS, D. and DIACONIS, P. (1986). Shuffling cards and stopping times. *Amer. Math. Monthly* **93** 333–348.

[2] ALDOUS, D. and DIACONIS, P. (1987). Strong uniform times and finite random walks. *Adv. in Appl. Math.* **8** 69–97.

[3] ALDOUS, D. and FILL, J. A. (1998). *Reversible Markov Chains and Random Walks on Graphs.* Unpublished manuscript. First draft of manuscript available via `http://www.stat.berkeley.edu/users/aldous`.

[4] ASMUSSEN, S., GLYNN, P. W. and THORISSON, H. (1992). Stationary detection in the initial transient problem. *ACM Trans. Modelling and Comput. Sim.* **2** 130–157.

[5] BESAG, J. E. (1986). On the statistical analysis of dirty pictures. *J. Roy. Statist. Soc. Ser. B* **48** 259–302.

[6] BESAG, J. E. and CLIFFORD, P. (1989). Generalized Monte Carlo significance tests. *Biometrika* **76** 633–642.

[7] BESAG, J. E. and CLIFFORD, P. (1991). Sequential Monte Carlo p-values. *Biometrika* **78** 301–304.

[8] BESAG, J., GREEN, P., HIGDON, D. and MENGERSEN, K. (1995). Bayesian computation and stochastic systems. *Statist. Sci.* **10** 3–66.

[9] BOROVKOV, A. A. and FOSS, S. G. (1992). Stochastically recursive sequences and their generalizations. *Siberian Adv. Math.* **2** 16–81.

[10] BOROVKOV, A. A. and FOSS, S. G. (1994). Two ergodicity criteria for stochastically recursive sequences. *Acta Appl. Math.* **34** 125–134.

[11] COWEN, L. J. (1996). Personal communication.

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

[13] DIACONIS, P. and FILL, J. A. (1990). Strong stationary times via a new form of duality. *Ann. Probab.* **18** 1483–1522.

[14] DIACONIS, P. and SALOFF-COSTE, L. (1995). What do we know about the Metropolis algorithm? In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing* 112–129.

[15] DIACONIS, P. and SALOFF-COSTE, L. (1995). Geometry and randomness. Unpublished lecture notes.

[16] FILL, J. A. (1998). The move-to-front rule: a case study for two exact sampling algorithms. *Probab. Eng. Info. Sci.* To appear.

[17] FILL, J. A. and MACHIDA, M. (1998). Stochastic monotonicity and realizable monotonicity. Unpublished notes.

[18] FILL, J. A. and MACHIDA, M. (1997). Duality relations for Markov chains on partially ordered state spaces. Unpublished notes.

[19] FOSS, S. G. (1983). On ergodicity conditions in multi-server queues. *Siberian Math. J.* **24** 168–175.

[20] FOSS, S., TWEEDIE, R. L. and CORCORAN, J. (1997). Simulating the invariant measures of Markov chains using backward coupling at regeneration times. Preprint.

[21] GEMAN, S. and GEMAN, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-6**(6).

[22] HÄGGSTRÖM, O. and NELANDER, K. (1997). Exact sampling from anti-monotone systems. Preprint.

[23] HÄGGSTRÖM, O., VAN LIESHOUT, M. N. M. and MØLLER, J. (1996). Characterisation results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes. Preprint.

[24] JOHNSON, V. E. (1996). Studying convergence of Markov chain Monte Carlo algorithms using coupled sample paths. *J. Amer. Statist. Assoc.* **91** 154–166.

[25] KAMAE, T., KRENGEL, U. and O'BRIEN, G. L. (1977). Stochastic inequalities on partially ordered state spaces. *Ann. Probab.* **5** 899–912.

[26] KENDALL, W. S. (1996). Perfect simulation for the area-interaction point process. In *Probability Perspective* (L. Accardi and C. C. Heyde, eds.). World Scientific Press, Singapore. To appear.

[27] KENDALL, W. S. (1996). On some weighted Boolean models. In *Advances in Theory and Application of Random Sets* (D. Jeulin, ed.) 105–120. World Scientific Press, Singapore.

[28] LIGGETT, T. (1985). *Interacting Particle Systems.* Springer, New York.

[29] LOVÁSZ, L. and WINKLER, P. (1995). Exact mixing in an unknown Markov chain. *Electron. J. Combin.* **2** R15.

[30] LUND, R. B., WILSON, D. B., FOSS, S. and TWEEDIE, R. L. (1997). Exact and approximate simulation of the invariant measures of Markov chains. Preprint.

[31] PROPP, J. G. and WILSON, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms* **9** 223–252.

[32] ROSS, S. (1994). *A First Course in Probability*, 4th ed. Macmillan, New York.

[33] SINCLAIR, A. (1993). *Algorithms for Random Generation and Counting: A Markov Chain Approach.* Birkhäuser, Boston.

[34] SPERNER, E. (1928). Ein Satz über Untermengen einer endlichen Menge. *Math. Z.* **27** 544–548.

[35] STANLEY, R. P. (1986). *Enumerative Combinatorics* **1**. Wadsworth & Brooks/Cole, Monterey, CA.

[36] STRASSEN, V. (1965). The existence of probability measures with given marginals. *Ann. Math. Statist.* **36** 423–439.

[37] WILSON, D. B. and PROPP, J. G. (1997). How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *J. Algorithms*. To appear.

[38] WILSON, D. B. (1995). Personal communication.

DEPARTMENT OF MATHEMATICAL SCIENCES
JOHNS HOPKINS UNIVERSITY
BALTIMORE, MARYLAND 21218-2682
E-MAIL: jimfill@jhu.edu