Fields Institute Communications
Volume **00**, 0000

Extension of Fill's perfect rejection sampling algorithm to general chains (EXT. ABS.)

James Allen Fill
Department of Mathematical Sciences
The Johns Hopkins University
Baltimore, MD 21218–2682, U. S. A.
jimfill@jhu.edu

Motoya Machida
Department of Mathematical Sciences
The Johns Hopkins University
Baltimore, MD 21218–2682, U. S. A.
machida@mts.jhu.edu

Duncan J. Murdoch
Department of Statistical and Actuarial Sciences
University of Western Ontario
London, Ontario N6G 2E9, Canada
murdoch@fisher.stats.uwo.ca

Jeffrey S. Rosenthal
Department of Statistics
University of Toronto
Toronto, Ontario M5S 3G3, Canada
jeff@math.toronto.edu

Abstract. We provide an extension of the perfect sampling algorithm of Fill (1998) to general chains, and describe how use of bounding processes can ease computational burden. Along the way, we unearth a simple connection between the Coupling From The Past (CFTP) algorithm originated by Propp and Wilson (1996) and our extension of Fill's algorithm.

1 Introduction

Markov chain Monte Carlo (MCMC) methods have become extremely popular for Bayesian inference problems (consult, e.g., Gelfand and Smith [**16**], Smith

1991 *Mathematics Subject Classification.* Primary 60J10, 68U20; Secondary 60G40, 62D05, 62E25.

The first and second authors have been supported in part by NSF grants DMS–9626756 and DMS–9803780, and by the Acheson J. Duncan Fund for the Advancement of Research in Statistics.

The third and fourth authors have been supported in part by NSERC.

and Roberts [**36**], Tierney [**38**], Gilks et al. [**17**]), and for problems in other areas, such as spatial statistics, statistical physics, and computer science (see, e.g., Fill [**11**] or Propp and Wilson [**32**] for pointers to the literature) as a way of sampling approximately from a complicated unknown probability distribution π. An MCMC algorithm constructs a Markov chain with one-step transition kernel K and stationary distribution π; if the chain is run long enough, then under reasonably weak conditions (cf. Tierney [**38**]) it will converge in distribution to π, facilitating approximate sampling.

One difficulty with these methods is that it is difficult to assess convergence to stationarity. This necessitates the use of difficult theoretical analysis (e.g., Meyn and Tweedie [**27**], Rosenthal [**35**]) or problematic convergence diagnostics (Cowles and Carlin [**5**], Brooks, et al. [**2**]) to draw reliable samples and do proper inference.

An interesting alternative algorithm, called *coupling from the past* (CFTP), was introduced by Propp and Wilson [**32**] (see also [**33**] and [**34**]) and has been studied and used by a number of authors (including Kendall [**23**], Møller [**28**], Murdoch and Green [**30**], Foss and Tweedie [**15**], Kendall and Thönnes [**25**], Corcoran and Tweedie [**4**], Green and Murdoch [**18**], Murdoch and Rosenthal [**31**]). By searching backwards in time until paths from all starting states have coalesced, this algorithm uses the Markov kernel K to sample *exactly* from π.

Another method of perfect simulation, for finite-state stochastically monotone chains, was proposed by Fill [**11**]. Fill's algorithm is a form of rejection sampling. This algorithm was later extended by Møller and Schladitz [**29**] and Thönnes [**37**] to non-finite chains, motivated by applications to spatial point processes. Fill's algorithm has the advantage over CFTP of removing the correlation between the length of the run and the returned value, which eliminates bias introduced by an impatient user or a system crash and so is "interruptible". However, it has been used only for stochastically monotone chains, making heavy use of the ordering of state space elements. In his paper, Fill [**11**] indicated that his algorithm could be suitably modified to allow for the treatment of "anti-monotone" chains and (see his Section 11.2) indeed to generic chains. In this extended abstract we present a version of Fill's algorithm for generic chains; we, too, will provide an explanation in terms of rejection sampling. We have strived to keep to the spirit of the talks presented at the Workshop on Monte Carlo Methods held at the Fields Institute for Research in Mathematical Sciences in Toronto in October, 1998 and make our results accessible to a broad audience. Technical details are provided in the full paper [**14**].

Following is our interruptible algorithm for generic chains. We discuss some of the terminology used and other details of the algorithm in Section 3.

Algorithm 1.1 Choose and fix a positive integer t, choose an initial state $\mathbf{X}_t$ from any distribution absolutely continuous with respect to π, and perform the following routine. Run the time-reversed chain $\widetilde{K}$ for t steps, obtaining $\mathbf{X}_t, \mathbf{X}_{t-1}, \ldots, \mathbf{X}_0$ in succession. Then, reversing the direction of time, generate (possibly dependent) Markov chains, one [say $\vec{\mathbf{Y}}(x) = (\mathbf{Y}_0(x), \ldots, \mathbf{Y}_t(x))$, with $\mathbf{Y}_0(x) = x$] evolving from each element x of the state space $\mathcal{X}$ and each with transition kernel K. All these trajectories are to be multivariately coupled *ex post facto* with the trajectory $\vec{\mathbf{X}} = (\mathbf{X}_0, \ldots, \mathbf{X}_t)$, which is regarded as a trajectory from K; in particular, $\vec{\mathbf{Y}}(\mathbf{X}_0) = \vec{\mathbf{X}}$. Finally, we check whether all the values $\mathbf{Y}_t(x)$, $x \in \mathcal{X}$, agree. If they do, we call this *coalescence*, and the value $\mathbf{X}_0$ is accepted as an observation from π. If not, then the value $\mathbf{X}_0$ is rejected and so the routine fails. We then start

the routine again with an independent simulation, perhaps with a fresh choice of t and $\mathbf{X}_t$, and repeat until the algorithm succeeds.

Here is a simple intuitive explanation of why the algorithm works correctly. Imagine (1) *starting* the construction with $\mathbf{X}_0 \sim \pi$ and independently (2) building all of the paths $\vec{\mathbf{Y}}(x)$ [including $\vec{\mathbf{X}} = \vec{\mathbf{Y}}(\mathbf{X}_0)$] *simultaneously*. Since determination of coalescence and the value of the coalesced paths at time t rely only on the second piece of randomness, conditionally given coalescence to $\mathbf{X}_t = z$ (for any z) we will still have $\mathbf{X}_0 \sim \pi$, as desired. The algorithm builds the randomness in a different order, starting from $\mathbf{X}_t$.

Remark 1.2 (a) Note that no assumption is made in Algorithm 1.1 concerning monotonicity or discreteness of the state space.

(b) See (3.1) for the definition of $\widetilde{K}$.

(c) To couple all of the trajectories $\vec{\mathbf{Y}}(x)$ *ex post facto* with the trajectory $\vec{\mathbf{X}}$ means first to devise a multivariate coupling for all the trajectories by means of a transition rule and then to employ that coupling conditionally having observed the single trajectory $\vec{\mathbf{Y}}(\mathbf{X}_0) = \vec{\mathbf{X}}$. Just how this is done is described in Section 3.

(d) If coalescence occurs, then of course the common value of $\mathbf{Y}_t(x)$, $x \in \mathcal{X}$, is the initial state $\mathbf{X}_t$.

(e) We have reversed the direction of time, and the roles of the kernels K and $\widetilde{K}$, compared to Fill [**11**]. Furthermore, Fill's original algorithm also incorporated a search for a good value of t by doubling the previous value of t until the first success. For the most part, we shall not address such issues, instead leaving the choice of t entirely up to the user; but see Section 5.2.

(f) We have included the technical absolute continuity restriction on the distribution of $\mathbf{X}_t$ to ensure correctness. In a typical application, one might know a density for π with respect to some measure λ (for example, $\lambda =$ Lebesgue measure) up to a normalizing constant. Then if, for example, that density is positive on the entire state space $\mathcal{X}$, one need only take $\mathbf{X}_t$ to have any distribution having density with respect to λ (for example, a normal distribution).

Also, if the state space is discrete, or if it is continuous and all probabilistic ingredients to the algorithm (such as the kernel K) are sufficiently smooth, then the user may choose $\mathbf{X}_t$ deterministically and arbitrarily.

As mentioned above, we will discuss the details of Algorithm 1.1 in Section 3. First, in Section 2, we motivate our algorithm in the context of a rather general rejection sampling framework. A more rigorous treatment may be found in the full paper [**14**].

In Section 4 we discuss how the computational burden of tracking all of the trajectories $\mathbf{Y}(x)$ can be eased by the use of coalescence detection events in general and bounding processes in particular; these processes take on a very simple form (see Section 4.3) when the state space is partially ordered and the transition rule employed is monotone. In the full paper we present a computationally efficient modification of Algorithm 1.1 that applies when K is assumed to be stochastically monotone. (As discussed in Fill and Machida [**13**] and Machida [**26**], this is a weaker assumption than that of the existence of a monotone transition rule.) When the state space is finite, the algorithm for the stochastically monotone case reduces to Fill's original algorithm. The full paper also discusses a "cross-monotone" generalization of stochastic monotonicity.

In Section 5 we compare Algorithm 1.1 and CFTP. We also demonstrate a simple connection between CFTP and an infinite-time-window version of Algorithm 1.1 (namely, Algorithm 5.1).

This extended abstract discusses only algorithms for generating a single observation from a distribution π of interest. In the full paper, we discuss various strategies for perfect generation of samples of arbitrary size from π.

The goal of this extended abstract is to describe how to apply Fill's perfect sampling algorithm to a broad spectrum of problems of real interest, not to develop any specific applications in detail. But an underlying theme here is that while our extended algorithm (Algorithm 1.1) tends to be computationally more intricate than CFTP (see Section 5.1), theoretically it is as broadly applicable as is CFTP. In this spirit, we will point to the applied perfect sampling literature at appropriate junctures, taking note of past applications of both CFTP and Fill's algorithm as examples. We hope that our extension of the latter algorithm will stimulate further research into this less-used alternative for perfect MCMC simulation.

A valuable background resource is the annotated bibliography of perfect sampling maintained by Wilson [**39**].

2 Rejection sampling using auxiliary randomness

Given a bivariate distribution $\mathcal{L}(X, X')$, suppose that, for each x', we can simulate X from the conditional distribution $\mathcal{L}(X|X' = x')$. How can we simulate X from its marginal distribution $\pi := \mathcal{L}(X)$? This is the problem confronting us in the context of Algorithm 1.1, with $(X, X') := (\mathbf{X}_0, \mathbf{X}_t)$ and $\mathbf{X}_t$ chosen according to π. Indeed, we assumed there that the user can simulate from $\mathcal{L}(\mathbf{X}_0 \,|\, \mathbf{X}_t = x') = \widetilde{K}^t(x', \cdot)$, where $\widetilde{K}^t$ is the t-step time-reversal transition kernel; and, when $\mathbf{X}_t \sim \pi$, we have $\mathcal{L}(\mathbf{X}_0) = \pi$. Of course, if the user could simulate $\mathbf{X}_t \sim \pi$, then either $\mathbf{X}_t$ or $\mathbf{X}_0$ could be used directly as an observation from π. But, for MCMC, this is an unreasonable assumption.

So we turn to rejection sampling (e.g., Devroye [**6**]), done conditionally given $X' = x'$. Our goal is to use the observation x from $\mathcal{L}(X|X' = x')$ to simulate one from π. This can be done by accepting x as an observation from π with probability

$$\gamma_{x',x} := \alpha_{x'} \frac{\pi(dx)}{P(X \in dx | X' = x')}$$

for any $\alpha_{x'}$ chosen to make $0 < \gamma_{x',x} \le 1$; indeed, then

$$P(X \in dx \,|\, X' = x', \text{accept}) = \frac{P(X \in dx, \text{accept} \,|\, X' = x')}{P(\text{accept} \,|\, X' = x')} = \frac{\alpha_{x'} \pi(dx)}{\alpha_{x'} \int \pi(dy)} = \pi(dx).$$

The question remains how to engineer a coin-flip with probability $\gamma_{x',x}$ of heads, given that one can rarely *compute* $\gamma_{x',x}$ in practice.

However, if we can find an event C so that

$$P(C \,|\, X' = x', X = x) = \gamma_{x',x} \quad \text{for all } x', x, \tag{2.1}$$

then we need only accept when (and only when) C occurs. Condition (2.1) requires precisely that

$$P(\{X' \in B'\} \cap C \cap \{X \in B\}) = \pi(B) \int_{B'} \alpha_{x'} P(X' \in dx') \quad \text{for all } B, B'. \tag{2.2}$$

Note that if we can choose C so that condition (2.2) holds, then setting B to be the entire state space $\mathcal{X}$ for X, we obtain

$$P(\{X' \in B'\} \cap C) = \int_{B'} \alpha_{x'} P(X' \in dx') \quad \text{for all } B'$$

and hence

$$P(\{X' \in B'\} \cap C \cap \{X \in B\}) = \pi(B)\, P(\{X' \in B'\} \cap C) \quad \text{for all } B, B'. \quad (2.3)$$

Conversely, if we can choose C so that condition (2.3) holds, then we can choose

$$\alpha_{x'} := P(C \,|\, X' = x')$$

to satisfy (2.1) and (2.2).

How might we choose C to satisfy (2.3)? Observe that if C and another variable Y are such that (i) $Y = X'$ over the event C and (ii) X and the pair (Y, C) are independent, then

$$\begin{aligned} \text{LHS(2.3)} &= P(\{X' \in B'\} \cap C \cap \{X \in B\}) = P(\{Y \in B'\} \cap C \cap \{X \in B\}) \\ &= P(\{Y \in B'\} \cap C)\, \pi(B). \end{aligned}$$

Setting $B = \mathcal{X}$ and then substituting, we obtain (2.3). The Algorithm 1.1 is designed precisely so that we may take $(Y, C) := (\mathbf{Y}_t(x_0^*), \{\text{coalescence}\})$ for an arbitrarily chosen (but fixed) $x_0^* \in \mathcal{X}$ to satisfy (i) and (ii). This is discussed further in Section 3.1 below.

3 Details for Algorithm 1.1

3.1 The ingredients. The goal of this subsection is to describe in some detail how to apply Algorithm 1.1.

The space $(\mathcal{X}, \mathcal{B})$: It is sufficient that $(\mathcal{X}, \mathcal{B})$ be a complete separable metric space. In particular, this covers the case that $\mathcal{X}$ is discrete or Euclidean. See Section 3.1 of the full paper [**14**] for further discussion.

The kernel K and its time-reversal $\widetilde{K}$: Let K be a Markov transition kernel on $\mathcal{X}$. The kernel is chosen (by the user) so that π is a stationary distribution, i.e., so that

$$\int_{\mathcal{X}} \pi(dx) K(x, dy) = \pi(dy) \quad \text{on } \mathcal{X}.$$

The time-reversed kernel $\widetilde{K}$ (also on $\mathcal{X}$) is then defined by

$$\pi(dx) K(x, dy) = \pi(dy) \widetilde{K}(y, dx) \quad \text{on } \mathcal{X} \times \mathcal{X}. \quad (3.1)$$

The transition rule ϕ: There exists a transition rule which can be used to drive the construction of the Markov chain of interest. More precisely, there exists a probability space $(\mathcal{U}, \mathcal{F}, \mu)$ and a (suitably measurable) function $\phi : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ such that

$$K(x, B) = \mu\{u : \phi(x, u) \in B\}, \quad x \in \mathcal{X}, \quad B \in \mathcal{B}. \quad (3.2)$$

Such ϕ (with accompanying μ) is sometimes called a *transition rule*. We choose and fix such a (ϕ, μ).

Remark 3.1 (a) A transition rule ϕ can always be found that uses $(\mathcal{U}, \mathcal{F}, \mu) = ([0,1], \text{Borels}, \text{uniform distribution})$. In the special case that $\mathcal{X}$ is the unit interval, we can in fact use

$$\phi(x, u) \equiv G(x, u)$$

where $G(x,\cdot)$ is the usual inverse probability transform corresponding to the distribution function $u \mapsto K(x,[0,u])$.

(b) If $\mathcal{X}$ is discrete (finite or countably infinite), a very simple alternative choice is the following "independent-transitions" transition rule. Let $\mathcal{U} = \mathcal{X}^{\mathcal{X}}$, let μ be product measure with xth marginal $K(x,\cdot)$ ($x \in \mathcal{X}$), and let ϕ be the evaluation function

$$\phi(x,u) := u(x).$$

(c) Many interesting examples of transition rules can be found in the literature, including Diaconis and Freedman [**8**] and the references cited in Section 1.

(d) Usually there is a wealth of choices of transition rule, and the art is to find one giving rapid and easily detected coalescence. Without going into details at this point, we remark that the transition rule in (b) usually performs quite badly, while transition rules having a certain monotonicity property will perform well under monotonicity assumptions on K.

The Markov chain and a first probability space: We will need to set up two probability spaces. The first space $(\Omega, \mathcal{A}, P)$—designated in ordinary typeface—will be useful for theoretical considerations and for the computation of certain conditional probability distributions. The second space $(\mathbf{\Omega}, \mathbf{A}, \mathbf{P})$—designated in boldface type—will be the probability space actually simulated when the algorithm is run. All random variables defined on the first space (respectively, second space) will also be designated in ordinary typeface (resp., boldface type). We have chosen this notational system to aid the reader: Corresponding variables, such as X_0 and $\mathbf{X}_0$, will play analogous roles in the two spaces.

From our previous comments it is easy to see that there exists a space $(\mathcal{U}, \mathcal{F})$, a transition rule (ϕ, μ), and a probability space $(\Omega, \mathcal{A}, P)$ on which are defined independent random variables $X_0, U_1, U_2, \ldots, U_t$ with $X_0 \sim \pi$ and each $U_s \sim \mu$. Now inductively define

$$X_s := \phi(X_{s-1}, U_s), \quad 1 \le s \le t. \tag{3.3}$$

Then $\vec{X} := (X_0, \ldots, X_t)$ is easily seen to be a stationary Markov chain with kernel K, in the sense that

$$P(X_0 \in dx_0, \ldots, X_t \in dx_t) = \pi(dx_0)K(x_0, dx_1)\cdots K(x_{t-1}, dx_t) \quad \text{on } \mathcal{X}^{t+1}. \tag{3.4}$$

In fact, for each $x \in \mathcal{X}$ we obtain a chain with kernel K started from x by defining $Y_0(x) := x$ and, inductively,

$$Y_s(x) := \phi(Y_{s-1}(x), U_s).$$

Let $\vec{Y}(x) := (Y_0(x), \ldots, Y_t(x))$. In this notation we have $\vec{Y}(X_0) = \vec{X}$. Let

$$C := \{Y_t(x) \text{ does not depend on } x\} \tag{3.5}$$

denote the event that the trajectories $\vec{Y}(x)$ have all coalesced by time t. Fixing $x_0^* \in \mathcal{X}$ arbitrarily and taking

$$X = X_0, \quad X' = X_t, \quad Y = Y_t(x_0^*), \quad C \text{ as at (3.5)}$$

to match up with the notation in Section 2, key observations are that, for any (measurable) subset B of $\mathcal{X}$, $Y = X'$ over the event C, and X and the event $\{Y \in B\} \cap C$ are independent. The independence here follows from the fact that X_0 and $\vec{U} := (U_1, \ldots, U_t)$ have been chosen to be independent.

A second probability space and the algorithm: We make use of the auxiliary randomness provided by $X_1, \ldots, X_{t-1}$ and $\vec{U}$ and compute conditional probability distributions for the first probability space in stages. First observe from (3.4) and repeated use of (3.1) that

$$P(X_0 \in dx_0, \ldots, X_{t-1} \in dx_{t-1} \,|\, X_t = x_t) = \widetilde{K}(x_t, dx_{t-1}) \cdots \widetilde{K}(x_1, dx_0).$$

Next, we will discuss in Section 3.2 how to compute $\mathcal{L}(\vec{U} \,|\, \vec{X} = \vec{x})$. Finally, whether or not C occurs is determined solely by the randomness in $\vec{U}$, so the conditional probability of C given $(\vec{X}, \vec{U}) = (\vec{x}, \vec{u})$ is degenerately either 0 or 1.

Moreover, our discussion has indicated how to set up and *simulate* the second space. As discussed in Section 1, we assume that the user knows enough about π qualitatively to be able to simulate $\mathbf{X}_t$ so that $\mathcal{L}_{\mathbf{P}}(\mathbf{X}_t) \ll \pi$. Having chosen $\mathbf{X}_t = x_t$, the user draws an observation $\mathbf{X}_{t-1} = x_{t-1}$ from $\widetilde{K}(x_t, \cdot)$, then an observation $\mathbf{X}_{t-2} = x_{t-2}$ from $\widetilde{K}(x_{t-1}, \cdot)$, etc. Next, having chosen $\vec{\mathbf{X}} = \vec{x}$ [i.e., $(\mathbf{X}_0, \ldots, \mathbf{X}_t) = (x_0, \ldots, x_t)$], the user draws an observation $\vec{\mathbf{U}} = \vec{u}$ from $\mathcal{L}(\vec{U} \,|\, \vec{X} = \vec{x})$ and constructs $\vec{\mathbf{Y}}(x) = (\mathbf{Y}_0(x), \ldots, \mathbf{Y}_t(x))$ by setting $\mathbf{Y}_0(x) := x$ and, inductively,

$$\mathbf{Y}_s(x) := \phi(\mathbf{Y}_{s-1}(x), \mathbf{U}_s).$$

Finally, the user declares that $\mathbf{C}$, or *coalescence*, has occurred if and only if the values of $\mathbf{Y}_t(x)$, $x \in \mathcal{X}$, all agree. The conditional distribution of output from Algorithm 1.1 given that it ultimately succeeds (perhaps only after many iterations of the basic routine) is π, as desired.

Remark 3.2 (a) If $\mathbf{P}(\mathbf{C}) > 0$ for suitably large t, then ultimate success is (a.s.) guaranteed if the successive choices of t become large. A necessary condition for ultimate positivity of $\mathbf{P}(\mathbf{C})$ is uniform ergodicity of K. This condition is also sufficient, in the (rather weak) sense that if K is uniformly ergodic, then there exists a finite integer m and a transition rule ϕ_m for the m-step kernel K^m such that Algorithm 1.1, applied using ϕ_m, has $\mathbf{P}(\mathbf{C}) > 0$ when t is chosen sufficiently large. Compare the analogous Theorem 4.2 for CFTP in Foss and Tweedie [**15**].

(b) Just as discussed in Fill [**11**] (see especially the end of Section 7 there), the algorithm (including its repetition of the basic routine) we have described is interruptible; that is, its running time (as measured by number of Markov chain steps) and output are independent random variables, conditionally given that the algorithm eventually terminates.

(c) If the user chooses the value of $\mathbf{X}_t$ ($= z$, say) deterministically, then all that can be said in general is that the algorithm works properly for π-a.e. such choice. In this case, let the notation $\mathbf{P}_z(\mathbf{C})$ reflect the dependence of $\mathbf{P}(\mathbf{C})$ on the initial state z. Then clearly

$$\int \mathbf{P}_z(\mathbf{C})\, \pi(dz) = P(C),$$

which is the unconditional probability of coalescence in our first probability space and therefore equal to the probability that CFTP terminates over an interval of width t. This provides a first link between CFTP and Algorithm 1.1. (Very) roughly recast, the distribution of running time for CFTP is the stationary mixture, over initial states, of the distributions of running time for Algorithm 1.1. For further elaboration of the connection between the two algorithms, see Section 5.3.

3.2 Imputation. In order to be able to run Algorithm 1.1, the user needs to be able to impute $\vec{U}$ from $\vec{X}$, i.e., to draw from $\mathcal{L}(\vec{U} \,|\, \vec{X} = \vec{x})$. In this subsection we explain how to do this. We begin with the computation

$$\begin{aligned}
&P(\vec{U} \in d\vec{u} \,|\, \vec{X} = \vec{x}) \\
&\quad = \quad P(\vec{U} \in d\vec{u} \,|\, X_0 = x_0,\, \phi(x_0, U_1) = x_1,\, \ldots,\, \phi(x_{t-1}, U_t) = x_t) \quad \text{by (3.3)} \\
&\quad = \quad P(\vec{U} \in d\vec{u} \,|\, \phi(x_0, U_1) = x_1,\, \ldots,\, \phi(x_{t-1}, U_t) = x_t) \quad \text{by indep. of } X_0 \text{ and } \vec{U} \\
&\quad = \quad P(U_1 \in du_1 \,|\, \phi(x_0, U_1) = x_1) \times \cdots \times P(U_t \in du_t \,|\, \phi(x_{t-1}, U_t) = x_t) \\
&\qquad\qquad \text{by independence of } U_1, \ldots, U_t \\
&\quad = \quad P(U_1 \in du_1 \,|\, \phi(x_0, U_1) = x_1) \times \cdots \times P(U_1 \in du_t \,|\, \phi(x_{t-1}, U_1) = x_t) \\
&\qquad\qquad \text{since } U_1, \ldots, U_t \text{ are identically distributed} \\
&\quad = \quad P(U_1 \in du_1 \,|\, X_0 = x_0,\, X_1 = x_1) \times \cdots \times P(U_1 \in du_t \,|\, X_0 = x_{t-1},\, X_1 = x_t),
\end{aligned}$$

where the last equality is justified in the same fashion as for the first two. This establishes

Lemma 3.3 *The t-fold product of the measures*

$$P(U_1 \in du_1 \,|\, X_0 = x_0,\, X_1 = x_1),\ \ldots,\ P(U_1 \in du_t \,|\, X_0 = x_{t-1},\, X_1 = x_t)$$

serves as a conditional probability distribution $P(\vec{U} \in d\vec{u} \,|\, \vec{X} = \vec{x})$.

In setting up the second probability space, therefore, the user, having chosen $\vec{\mathbf{X}} = \vec{x}$, draws an observation $\vec{\mathbf{U}} = \vec{u}$ by drawing $\mathbf{U}_1, \ldots, \mathbf{U}_t$ independently, with $\mathbf{U}_s$ chosen according to the distribution $\mathcal{L}(U_1 \,|\, X_0 = x_{s-1},\, X_1 = x_s)$.

Remark 3.4 (a) If $\mathcal{X}$ is discrete, suppose we use the "independent-transitions" rule ϕ discussed in Remark 3.1(b). Then the measure μ, but with the x_0th marginal replaced by δ_{x_1}, serves as $\mathcal{L}(U_1 \,|\, \phi(x_0, U_1) = x_1) = \mathcal{L}(U_1 \,|\, U_1(x_0) = x_1)$ and therefore as $\mathcal{L}(U_1 \,|\, X_0 = x_0,\, X_1 = x_1)$. Informally stated, having chosen $\mathbf{X}_s = x_s$ and $\mathbf{X}_{s-1} = x_{s-1}$, the user imputes the forward-trajectory transitions from time $s-1$ to time s in Algorithm 1.1 by declaring that the transition from state x_{s-1} is to state x_s and that the transitions from other states are chosen independently according to their usual non-$\vec{X}$-conditioned distributions.

(b) As another example, suppose that $\mathcal{X} = [0, 1]$ and we use the inverse probability transform transition rule discussed in Remark 3.1(a). Suppose also that each distribution function $F(x_0, \cdot) = K(x_0, [0, \cdot])$ is strictly increasing and onto $[0, 1]$. Then $\delta_{F(x_0, x_1)}$ serves as $\mathcal{L}(U_1 \,|\, X_0 = x_0,\, X_1 = x_1)$. Informally stated, a generated pair $(\mathbf{X}_s, \mathbf{X}_{s-1}) = (x_s, x_{s-1})$ completely determines the value $F(x_{s-1}, x_s)$ for $\mathbf{U}_s$.

3.3 A toy example. We illustrate Algorithm 1.1 for a very simple example and two different choices of transition rule. Consider the discrete state space $\mathcal{X} = \{0, 1, 2\}$, and let π be uniform on $\mathcal{X}$. Let K correspond to simple symmetric random walk with holding probability $1/2$ at the endpoints; that is, putting $k(x, y) := K(x, \{y\})$,

$$\begin{aligned}
&k(0,0) = k(0,1) = k(1,0) = k(1,2) = k(2,1) = k(2,2) = 1/2, \\
&k(0,2) = k(1,1) = k(2,0) = 0.
\end{aligned}$$

The stationary distribution is π. As for any ergodic birth-and-death chain, K is reversible with respect to π, i.e., $\widetilde{K} = K$. Before starting the algorithm, choose a transition rule; this is discussed further below.

For utter simplicity of description, we choose $t = 2$ and (deterministically) $\mathbf{X}_t = 0$ (say); as discussed in Section 1, a deterministic start is permissible here. We then choose $\mathbf{X}_1 \sim K(0, \cdot)$ and $\mathbf{X}_0 \,|\, \mathbf{X}_1 \sim K(\mathbf{X}_1, \cdot)$. How we proceed from this juncture depends on what we chose for ϕ.

One choice is the independent-transitions rule discussed in Remarks 3.1(b) and 3.4(a). The algorithm's routine can then be run using 6 independent random bits: these decide $\mathbf{X}_1$ (given $\mathbf{X}_2$), $\mathbf{X}_0$ (given $\mathbf{X}_1$), and the 4 transitions in the second (forward) phase of the routine not already determined from the rule

$$\mathbf{X}_{s-1} \mapsto \mathbf{X}_s \text{ from time } s-1 \text{ to time } s \ (s = 1, 2).$$

There are thus a total of $2^6 = 64$ possible overall simulation results, each having probability $1/64$. We check that exactly 12 of these produce coalescence. Of these 12 accepted results, exactly 4 have $\mathbf{X}_0 = 0$, another 4 have $\mathbf{X}_0 = 1$, and a final 4 have $\mathbf{X}_0 = 2$. Thus $\mathbf{P}(\mathbf{C}) = 12/64 = 3/16$, and we confirm that $\mathcal{L}_{\mathbf{P}_\mathbf{C}}(\mathbf{X}_0) = \pi$, as should be true. An identical result holds if instead we choose $\mathbf{X}_t = 1$ or $\mathbf{X}_t = 2$.

An alternative choice adapts Remarks 3.1(a) and 3.4(b) to our discrete setting. Note that we can use $(\mathcal{U}, \mu) = (\{0, 1\}, \text{uniform})$ and

$$\phi(\cdot, u) = \begin{cases} \text{the mapping taking } \ 0,1,2 \text{ to } \ \ 0,0,1, & \text{respectively, if } u = 0 \\ \text{the mapping taking } \ 0,1,2 \text{ to } \ \ 1,2,2, & \text{respectively, if } u = 1. \end{cases}$$

Choosing $t = 2$ and $\mathbf{X}_t = 0$ as before, the algorithm can now be run with just 2 random bits. In this case we check that exactly 3 of the 4 possible simulation results produce coalescence, 1 each yielding $\mathbf{X}_0 = 0, 1, 2$. Note that $\mathbf{P}(\mathbf{C}) = 3/4$ is much larger for this choice of ϕ. In fact, since ϕ is a monotone transition rule (see Definition 4.2 in Fill [**11**] or Definition 4.3 below), for the choice $\mathbf{X}_t = 0$ it gives the highest possible value of $\mathbf{P}(\mathbf{C})$ among all choices of ϕ: see Remark 9.3(e) in Fill [**11**]. It also is a best choice when $\mathbf{X}_t = 2$. [On a minor negative note, we observe that $\mathbf{P}(\mathbf{C}) = 0$ for the choice $\mathbf{X}_t = 1$. Also note that the $\pi = (1/3, 1/3, 1/3)$-average of the acceptance probabilities $(3/4, 0, 3/4)$, namely, $1/2$, is the probability that forward coupling (or CFTP) done with the same transition rule gives coalescence within 2 time units; this corroborates Remark 3.2(c).]

Remark 3.5 Both choices of ϕ are easily extended to handle simple symmetric random walk on $\{0, \ldots, n\}$ for any n; the second (monotone) choice is again best possible. (We assume $\mathbf{X}_t = 0$.) For fixed $c \in (0, \infty)$ and large n, results in Fill [**11**] and Section 4 of Diaconis and Fill [**7**] imply that, for $t = cn^2$, the routine's success probability is approximately $p(c)$; here $p(c)$ increases smoothly from 0 to 1 as c increases from 0 to ∞. We have not attempted the corresponding asymptotic analysis for the independent-transitions rule. Of course our chain is only a "toy" example anyway, because direct sampling from π is elementary.

4 Coalescence detection and bounding processes

4.1 Conservative detection of coalescence and detection processes. Even for large finite state spaces $\mathcal{X}$, determining exactly whether or not coalescence occurs in Algorithm 1.1 can be prohibitively expensive computationally; indeed, in principle this requires tracking each of the trajectories $\vec{Y}(x)$, $x \in \mathcal{X}$ to completion. However, observe that if we repeat the development of Sections 3.1–3.2, replacing the coalescence event $\{Y_t(x) \text{ does not depend on } x\}$ of (3.5) by any *subset* C of this event whose occurrence (or not) can still be determined solely from $\vec{U}$, then everything goes through as before. We call such an event C a *coalescence detection event*

and reiterate that C is a conservative indication of coalescence: the occurrence of a given coalescence detection event is sufficient, but not necessary, for the occurrence of coalescence of the paths $\vec{Y}(x)$.

In practice, a coalescence detection event is constructed in terms of a *detection process*. What we mean by this is a stochastic process $\vec{D} = (D_0, \ldots, D_t)$, defined on the same probability space $(\Omega, \mathcal{A}, P)$ as $\vec{U}$ and $\vec{X}$, together with a subset Δ of its state space $\mathcal{D}$, such that

(a) $\vec{D}$ is constructed from $\vec{U}$, and
(b) $\{D_s \in \Delta \text{ for some } s \le t\} \subseteq \{Y_t(x) \text{ does not depend on } x\}$.

Then $C := \{D_s \in \Delta \text{ for some } s \le t\}$ is a coalescence detection event.

Remark 4.1 In practice, $\vec{D}$ usually evolves Markovianly using $\vec{U}$; more precisely, it is typically the case that there exists deterministic $d_0 \in \mathcal{D}$ and $\delta : \mathcal{D} \times \mathcal{U} \to \mathcal{D}$ such that $D_0 = d_0$ and [paralleling (3.3)]

$$D_s := \delta(D_{s-1}, U_s), \quad 1 \le s \le t.$$

The important consequence is that, having determined the trajectory $\vec{\mathbf{X}}$ and the imputed $\vec{\mathbf{U}}$, the user need only follow a single trajectory in the forward phase of the routine, namely, that of $\vec{D}$ (or rather its analogue $\vec{\mathbf{D}}$ in the simulated probability space).

Example 4.2 We sketch two illustrative examples of the use of detection processes that do not immediately fall into the more specific settings of Sections 4.2 or 4.3. We hasten to point out, however, that because of the highly special structure of these two examples, efficient implementation of Algorithm 1.1 avoids the use of the forward phase altogether; this is discussed for example (a) in Fill [**12**].

(a) Our first example is provided by the move-to-front (MTF) rule studied in [**12**]. Let K be the Markov kernel corresponding to MTF with independent and identically distributed record requests corresponding to probability weight vector $(w_1, \ldots, w_n)$; see (2.1) of [**12**] for specifics. The arguments of Section 4 of [**12**] show that if D_s is taken to be the set of all records requested at least once among the first s requests and Δ is taken to consist of all $(n-1)$-element subsets of the records $1, \ldots, n$, then $\vec{D}$ is a detection process. Similar detection processes can be built for the following generalizations of MTF: move-to-root for binary search trees (see Dobrow and Fill [**9**] [**10**]) and MTF-like shuffles of hyperplane arrangement chambers and more general structures (see Bidigare, et al. [**1**] and Brown and Diaconis [**3**]).

(b) A second example of quite similar spirit is provided by the (now well-known) Markov chain (X_t) for generating a random spanning arborescence of the underlying weighted directed graph, with vertex set $\mathcal{U}$, of a Markov chain (U_t) with kernel q. Consult Propp and Wilson [**34**] (who also discuss a more efficient "cycle-popping" algorithm) for details. We consider here only the special case that (U_t) is an i.i.d. sequence, i.e., that $q(v, w) \equiv q(w)$. A transition rule ϕ for the chain (X_t) is created as follows: for vertex u and arborescence x with root r, $\phi(x, u)$ is the arborescence obtained from x by adding an arc from r to u and deleting the unique arc in x whose tail is u. Then it can be shown that if D_s is taken to be the set of all vertices appearing at least once in $(U_1, \ldots, U_s)$ and $\Delta := \{\mathcal{U}\}$, then $\vec{D}$ is a detection process.

4.2 Bounding processes. We obtain a natural example of a detection process $\vec{D}$ when (a) $\vec{D}$ is constructed from $\vec{U}$, (b) the corresponding state space $\mathcal{D}$ is some collection of subsets of $\mathcal{X}$, with

$$\Delta := \{\{x'\} : \ x' \in \mathcal{X}\},$$

and

$$\text{(c)} \quad D_s \supseteq \{Y_s(x) : \ x \in \mathcal{X}\}.$$

The concept is simple: in this case, each set D_s is just a "conservative estimate" (i.e., a superset) of the corresponding set $\{Y_s(x) : x \in \mathcal{X}\}$ of trajectory values; thus if $D_s = \{x'\}$, then the trajectories $\vec{Y}(x)$ are coalesced to state x' at time s and remain coalesced thereafter. We follow the natural impulse to call such a set-valued detection process a *bounding process*. Such bounding processes arise naturally in the contexts of monotone and anti-monotone transition rules (and have been used by many authors): see the next subsection. Other examples of bounding processes can be found in works of Huber: see [**20**] and [**21**] in connection with CFTP and [**22**] in connection with our algorithm.

Of course, nothing is gained, in comparison to tracking all the trajectories, by the use of a bounding process unless the states of $\mathcal{D}$ have more concise representations than those of generic subsets of $\mathcal{X}$; after all, we could always choose $\mathcal{D} = 2^{\mathcal{X}}$ and $D_s = \{Y_s(x) : x \in \mathcal{X}\}$. One rather general setting where compact representations are often possible, discussed in the next subsection, is that of a partially ordered set (poset) $\mathcal{X}$.

4.3 Monotone transition rules. We now suppose that $\mathcal{X}$ is equipped with a partial order. We also assume here that there exist (necessarily unique) elements $\hat{0}$ and $\hat{1}$ in $\mathcal{X}$ (called *bottom element* and *top element*, respectively) such that $\hat{0} \leq x \leq \hat{1}$ for all $x \in \mathcal{X}$. We will discuss the case of monotone transition rules, where we can build from $\vec{U}$ a bivariate process $((L_s, V_s))$, taking values at each time s in $\mathcal{X} \times \mathcal{X}$, such that

$$L_s \leq Y_s(x) \leq V_s \quad \text{for all } 0 \leq s \leq t \text{ and all } x \in \mathcal{X}. \tag{4.1}$$

Then $D_s := [L_s, V_s] = \{x \in \mathcal{X} : \ L_s \leq x \leq V_s\}$ gives a bounding process, and the pair (L_s, V_s) is a quite concise representation of D_s.

Recall that our construction (3.3) of the Markov chain $\vec{X}$ with kernel K relies on the choice of a transition rule (ϕ, μ) satisfying (3.2).

Definition 4.3 *A transition rule ϕ is said to be* monotone *if each of the mappings*
$\phi(\cdot, u) : \mathcal{X} \to \mathcal{X}, \ u \in \mathcal{U}$, *is monotone increasing, i.e., if*

$$\phi(x, u) \leq \phi(y, u) \text{ for all } u \in \mathcal{U}$$

whenever $x \leq y$.

Suppose now that ϕ is monotone. Set $(L_0, V_0) := (\hat{0}, \hat{1})$ and, inductively,

$$(L_s, V_s) := (\phi(L_{s-1}, U_s), \phi(V_{s-1}, U_s)), \quad 1 \leq s \leq t.$$

One immediately verifies by induction that (4.1) is satisfied. Note that (L_s, V_s) is determined solely by $\vec{U}$ (as is the coalescence detection event $C = \{L_t = V_t\}$) and is nothing more than $(Y_s(\hat{0}), Y_s(\hat{1}))$. In plain language, since monotonicity is preserved, when the chains $\vec{Y}(\hat{0})$ and $\vec{Y}(\hat{1})$ have coalesced, so must have every $\vec{Y}(x)$.

Remark 4.4 (a) Lower and upper bounding processes can also be constructed when Algorithm 1.1 is applied with a so-called "anti-monotone" transition rule; we omit the details. See Häggström and Nelander [**19**], Huber [**21**], Kendall [**23**], Møller [**28**], Møller and Schladitz [**29**], and Thönnes [**37**] for further discussion in various specialized settings. There are at least two neat tricks associated with anti-monotone rules. The first is that, by altering the natural partial order on $\mathcal{X}$, such rules can be regarded, in certain bipartite-type settings, as monotone rules, in which case the performance analysis in Section 5.3 of [**14**] is available: consult Section 3 of [**19**], the paper [**29**], and Definition 5.1 in [**37**]. The second is that the poset $\mathcal{X}$ is allowed to be "upwardly unbounded" and so need not have a $\hat{1}$: consult Section 2 of [**28**] and, again, [**29**] and [**37**].

(b) Dealing with monotone rules on partially ordered state spaces without $\hat{1}$ is problematic and requires the use of "dominating processes." We comment that a dominating process provides a sort of *random* bounding process and is useful when the state space is noncompact, but we shall not pursue these ideas any further here. See Kendall [**23**] and Kendall and Møller [**24**] in the context of CFTP; we hope to discuss the use of dominating processes for our algorithm in future work.

5 Our algorithm and CTFP: comparison and connection

5.1 Comparison. How does our extension of Fill's algorithm, as given by Algorithm 1.1 and discussed in detail in Section 3, compare to CFTP? As we see it, our algorithm has two main advantages and one main disadvantage.

Advantages: As discussed in Section 1 and Remark 3.2(b) and in [**11**], a primary advantage of our algorithm is interruptibility. Given the close connection between the algorithms described in Section 5.3, one may reasonably view the extra computational costs of our algorithm (see "*Disadvantage*" below) as the costs of securing interruptibility. A related second advantage concerns memory allocation. Suppose, for example, that our state space $\mathcal{X}$ is finite and that each time-step of Algorithm 1.1, including the necessary imputation (recall Section 3.2), can be carried out using a bounded amount of memory. Then, for fixed t, our algorithm can be carried out using a fixed finite amount of memory. Unfortunately, it is rare in practice that the kernel K employed is sufficiently well analyzed that one knows in advance a value of t (and a value of the seed $\mathbf{X}_t$) giving a reasonably large probability $\mathbf{P}(\mathbf{C})$ of acceptance. Furthermore, the fixed amount of memory needed is in practice larger than the typical amount of memory allocated dynamically in a run of CFTP. Finally, we should note that Wilson [**40**] has very recently presented a version of CFTP which also can be carried out with a fixed finite amount of memory, and which does not require an *a priori* estimate of the mixing time of the chain.

Disadvantage: A major disadvantage of our Algorithm 1.1 concerns computational complexity. We refer the reader to [**11**] and [**12**] for a more detailed discussion in the setting of our Section 4.3 (and, more generally, the setting of stochastic monotonicity). Briefly, if no attention is paid to memory usage, our algorithm has running time competitive with CFTP: cf. Remark 3.2(c), and also the discussion in Remark 9.3(e) of [**11**] that the running time of our algorithm is, in a certain sense, best possible in the stochastically monotone setting. However, this analysis assumes that running time is measured in Markov chain steps; unfortunately, time-reversed steps can sometimes take longer than do forward steps to execute (e.g., [**12**]), and the imputation described in Section 3.2 is sometimes difficult to

carry out. Moreover, the memory usage for naive implementation of our algorithm can be exorbitant; how to trade off speed for reduction in storage needs is described in [**11**].

5.2 An alternative to Algorithm 1.1. Thus far we have been somewhat sketchy about the choice(s) of t in Algorithm 1.1. As discussed in Section 1, one possibility is to run the repetitions of the basic routine independently, doubling t at each stage. However, another possibility is to continue back in time, reusing the already imputed values $\mathbf{U}_s$ and checking again for coalescence. (There is an oblique reference to this alternative in Remark 9.3 of Fill [**11**].) This idea leads to the following algorithm.

Algorithm 5.1 Choose an initial state $\mathbf{X}_0 \sim \hat{\pi}$, where $\hat{\pi}$ is absolutely continuous with respect to π. Run the time-reversed chain $\widetilde{K}$, obtaining $\mathbf{X}_0, \mathbf{X}_{-1}, \dots$ in succession. Conditionally given $(\mathbf{X}_0, \mathbf{X}_{-1}, \dots) = (x_0, x_{-1}, \dots)$, generate independent random variables $\mathbf{U}_0, \mathbf{U}_{-1}, \dots$ with marginals

$$\mathbf{P}(\mathbf{U}_s \in du) = P(U \in du \,|\, \phi(x_{s-1}, U) = x_s), \quad s = 0, -1, \dots , \tag{5.1}$$

where, on the right, $\mathcal{L}_P(U) = \mu$ is given by (3.2). For $t = 0, 1, \dots$ and $x \in \mathcal{X}$, set $\mathbf{Y}_{-t}^{(-t)}(x) := x$ and, inductively,

$$\mathbf{Y}_s^{(-t)}(x) := \phi(\mathbf{Y}_{s-1}^{(-t)}(x), \mathbf{U}_s), \quad -t+1 \le s \le 0.$$

If $\mathbf{T} < \infty$ is the smallest t such that

$$\mathbf{Y}_0^{(-t)}(x),\ x \in \mathcal{X}, \text{ agree} \qquad (= \mathbf{X}_0), \tag{5.2}$$

then the algorithm succeeds and reports $\mathbf{W} := \mathbf{X}_{-\mathbf{T}}$ as an observation from π. Otherwise, the algorithm fails.

Remark 5.2 (a) We need only generate $\mathbf{X}_0, \mathbf{X}_{-1}, \dots , \mathbf{X}_{-t}$ and then impute $\mathbf{U}_0, \mathbf{U}_{-1}, \dots , \mathbf{U}_{-t+1}$ using (5.1) in order to check whether or not (5.2) holds. Thus if $\mathbf{T} < \infty$, then the algorithm terminates in finite time.

(b) We omit the detailed description à la Section 3. But the key in setting up the first probability space is *first* to choose $W \sim \pi$ and $U_0, U_{-1}, \dots$ all mutually independent and *then*, having determined the backwards coalescence time T from $U_0, U_{-1}, \dots$, to set $X_{-T} := W$.

(c) We may relax the condition that $\mathbf{T}$ be the *smallest* t satisfying (5.2), via the use of coalescence detection events as in Section 4. In particular, to save considerably on computational effort, we may let $\mathbf{T}'$ be the smallest t which is a power of 2 such that (5.2) holds and report $\mathbf{X}_{-\mathbf{T}'}$ instead.

(d) Algorithm 5.1, and likewise its variant in remark (c), is interruptible: $\mathbf{T}$ and $\mathbf{W}$ are conditionally independent given success.

(e) Uniform ergodicity of K is necessary (and, in a weak sense, sufficient) for almost sure success of Algorithm 5.1; cf. Remark 3.2(a).

5.3 Connection with CFTP. There is a strong and simple connection between CFTP and our Algorithm 5.1. Indeed, suppose we carry out the usual CFTP algorithm to sample from π, using kernel K, transition rule ϕ, and driving variables $\vec{U} = (U_0, U_{-1}, \dots)$. Let T denote the backwards coalescence time and let $X_0 \sim \pi$ denote the terminal state output by CFTP. Let $W \sim \pi$ independent of $\vec{U}$, and follow the trajectory from $X_{-T} := W$ to X_0; call this trajectory $\vec{X} = (X_{-T}, \dots , X_0)$. Since X_0 is determined solely by $\vec{U}$, the random variables W and X_0 are independent.

When $\hat{\pi} = \pi$ in Algorithm 5.1, the algorithm simply constructs the same probability space as for CFTP, but with the ingredients generated in a different chronological order: first $X_0, X_{-1}, \ldots$; then $\vec{U}$ (which determines T); then $W := X_{-T}$. Again $X_0 \sim \pi$ and $W \sim \pi$ are independent.

Remark 5.3 (a) Because of this statistical independence, it does not matter in Algorithm 5.1 that we actually use $\mathbf{X}_0 \sim \hat{\pi} \neq \pi$.

(b) The fact (1) that W, unlike X_0, is independent of $\vec{U}$, together with (2) that T depends solely on $\vec{U}$, explains why our algorithm is interruptible and CFTP is not.

(c) In a single run of CFTP, the user would of course be unable to choose $W \sim \pi$ as above, just as in a single run of Algorithm 5.1 we do not actually choose $X_0 \sim \pi$. So one might regard our described connection between the two algorithms as a bit metaphorical. But see Section 7.2 of [**14**].

References

[1] Bidigare, P., Hanlon, P., and Rockmore, D. *A combinatorial description of the spectrum for the Tsetlin library and its generalization to hyperplane arrangements.* Duke Math. J. (1997), to appear.

[2] Brooks, S. P., Dellaportas, P., and Roberts, G. O. *An approach to diagnosing total variation convergence of MCMC algorithms.* J. Comput. Graph. Statist. **6** (1997), 251–265.

[3] Brown, K. and Diaconis, P. *Random walk and hyperplane arrangements.* Ann. Probab. (1997), to appear.

[4] Corcoran, J. and Tweedie, R. L. *Perfect simulation of Harris recurrent Markov chains.* Preprint (1999). Colorado State University.

[5] Cowles, M. K. and Carlin, B. P. *Markov chain Monte Carlo convergence diagnostics: a comparative review.* J. Amer. Statist. Assoc. **91** (1996), 883–904.

[6] Devroye, L. *Nonuniform random variate generation*, Springer–Verlag, New York, 1986.

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

[8] Diaconis, P. and Freedman, D. *Iterated random functions.* SIAM Rev. **41** (1999), 45–76.

[9] Dobrow, R. P. and Fill, J. A. *On the Markov chain for the move-to-root rule for binary search trees.* Ann. Appl. Probab. **5** (1995), 1–19.

[10] Dobrow, R. P. and Fill, J. A. *Rates of convergence for the move-to-root Markov chain for binary search trees.* Ann. Appl. Probab. **5** (1995), 20–36.

[11] Fill, J. A. *An interruptible algorithm for perfect sampling via Markov chains.* Ann. Appl. Probab. **8** (1998), 131–162.

[12] Fill, J. A. *The move-to-front rule: a case study for two perfect sampling algorithms.* Probab. Engrg. Inform. Sci. **12** (1998), 283–302.

[13] Fill, J. A. and Machida, M. *Stochastic and realizable monotonicity.* Preprint (1998). Available from `http://www.mts.jhu.edu/~fill/`.

[14] Fill, J. A., Machida, M., Murdoch, D., and Rosenthal, J. *Extension of Fill's perfect rejection sampling algorithm to general chains.* Preprint (1999). Available from `http://www.mts.jhu.edu/~fill/`.

[15] Foss, S. G. and Tweedie, R. L. *Perfect simulation and backward coupling.* Comm. Statist. Stochastic Models **14** (1998), 187–203.

[16] Gelfand, A. E. and Smith, A. F. M. *Sampling-based approaches to calculating marginal densities.* J. Amer. Statist. Assoc. **85** (1990), 398–409.

[17] Gilks, W. R., Richardson, S., and Spiegelhalter, D. J., editors. *Markov Chain Monte Carlo in Practice*, Chapman and Hall, London, 1996.

[18] Green, P. J. and Murdoch, D. J. *Exact sampling for Bayesian inference: towards general purpose algorithms.* Preprint (1998).

[19] Häggström, O. and Nelander, K. *Exact sampling from anti-monotone systems.* Statist. Neerlandica **52** (1998), 360–380.

[20] Huber, M. *Efficient exact sampling from the Ising model using Swendsen–Wang.* Preprint (1998). A two-page version appears in Tenth Annual ACM-SIAM Symposium on Discrete Algorithms.

[21] Huber, M. *Exact sampling and approximate counting techniques*, Proceedings of the 30th ACM Symposium on the Theory of Computing, 1998, pp. 31–40.
[22] Huber, M. *Interruptible exact sampling and construction of strong stationary times for Markov chains.* Preprint (1998).
[23] Kendall, W. *Perfect simulation for the area-interaction point process.* Accardi, L. and Heyde, C. C., editors, Probability Towards 2000, Springer–Verlag, New York, 1998, pp. 218–234.
[24] Kendall, W. and Møller, J. *Perfect Metropolis–Hastings simulation of locally stable point processes.* Preprint (1999).
[25] Kendall, W. and Thönnes, E. *Perfect simulation in stochastic geometry.* Pattern Recognition (1998), special issue on random sets, to appear.
[26] Machida, M. *Stochastic monotonicity and realizable monotonicity*, Ph.D. dissertation, Department of Mathematical Sciences, The Johns Hopkins University, 1999. Available from `http://www.mts.jhu.edu/~machida/`.
[27] Meyn, S. P. and Tweedie, R. L. *Computable bounds for convergence rates of Markov chains.* Ann. Appl. Probab. **4** (1994), 981–1011.
[28] Møller, J. *Perfect simulation of conditionally specified models.* J. R. Stat. Soc. Ser. B **61** (1999), 251–264.
[29] Møller, J. and Schladitz, K. *Extensions of Fill's algorithm for perfect simulation.* J. R. Stat. Soc. Ser. B (1998), to appear.
[30] Murdoch, D. J. and Green, P. J. *Exact sampling from a continuous state space.* Scand. J. Statist. **25** (1998), 483–502.
[31] Murdoch, D. J. and Rosenthal, J. S. *Efficient use of exact samples.* Preprint (1998).
[32] Propp, J. G. and Wilson, D. B. *Exact sampling with coupled Markov chains and applications to statistical mechanics.* Random Structures Algorithms **9** (1996), 223–252.
[33] Propp, J. G. and Wilson, D. B. *Coupling from the past: a user's guide.* Aldous, D. and Propp, J. G., editors, Microsurveys in Discrete Probability, vol. 41 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Amer. Math. Soc., Providence, RI, 1998, pp. 181–192.
[34] Propp, J. G. and Wilson, D. B. *How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph.* J. Algorithms **27** (1998), 170–217.
[35] Rosenthal, J. S. *Minorization conditions and convergence rates for Markov chain Monte Carlo.* J. Amer. Statist. Assoc. **90** (1995), 558–566.
[36] Smith, A. F. M. and Roberts, G. O. *Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods* (with discussion). J. R. Stat. Soc. Ser. B **55** (1993), 3–23.
[37] Thönnes, E. *Perfect simulation of some point processes for the impatient user.* Adv. in Appl. Probab. (1997), to appear.
[38] Tierney, L. *Markov chains for exploring posterior distributions* (with discussion). Ann. Statist. **22** (1994), 1701–1762.
[39] Wilson, D. B. *Annotated bibliography of perfectly random sampling with Markov chains.* Aldous, D. and Propp, J., editors, Microsurveys in Discrete Probability, vol. 41 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Amer. Math. Soc., Providence, RI, 1998, pp. 209–220. Updated versions are posted at `http://www.dbwilson.com/exact/`.
[40] Wilson, D. B. *How to couple from the past using a read-once source of randomness.* Preprint (1999). Available from `http://dbwilson.com/rocftp/`.