8

Perfection within Reach: Exact MCMC Sampling

Radu V. Craiu and Xiao-Li Meng

8.1 Intended Readership

The amount of research done by the Markov chain Monte Carlo (MCMC) community has been very impressive in the last two decades, as testified by this very volume. The power of MCMC has been demonstrated in countless instances in which more traditional numerical algorithms are helpless. However, one ubiquitous problem remains: the detection of convergence or lack thereof. Among the large number of procedures designed for detecting lack of convergence or for establishing convergence bounds (see, e.g., Chapters 6 and 7 in this volume), there is one class of MCMC algorithms that stands apart simply because it avoids the problem altogether. Whereas examples of such algorithms can be traced back to at least 1989 (see [56]), it is Propp and Wilson's 1996 seminal paper [48] that introduced the general scheme of *coupling from the past* (CFTP). Since then, there has been an intense search for perfect sampling or exact sampling algorithms, so named because such algorithms use Markov chains and yet obtain genuine independent and identically distributed (i.i.d.) draws—hence perfect or exact—from their limiting distributions within a finite number of iterations.

There is, of course, no free lunch. Whereas this is a class of very powerful algorithms, their construction and implementation tend to require a good deal of labor and great care. Indeed, even the most basic general themes are not entirely trivial to understand, and subtleties and traps can be overwhelming for novices. Our central goal in this chapter is therefore to provide an *intuitive* overview of some of the most basic sampling schemes developed since [48]. We do not strive for completeness, nor for mathematical generality or rigorousness. Rather, we focus on a few basic schemes and try to explain them as intuitively as we can, via figures and simple examples. The chapter is therefore not intended for the residents but rather the visitors of the MCMC kingdom who want to tour the magic land of perfect sampling. There are of course a number of other tour packages—see, for example, the list provided at http://dimacs.rutgers.edu/~dbwilson/exact.html, maintained by David Wilson. But we hope ours is one of the least expensive ones in terms of readers' mental investment, though by no means are we offering a free ride.

8.2 Coupling from the Past

8.2.1 Moving from Time-Forward to Time-Backward

The CFTP algorithm is based on an idea that is both simple and revolutionary. Suppose that we are interested in sampling from a distribution with probability law $\Pi(\cdot)$ with state space $\mathcal{S} \subset \mathbb{R}^d$. We define a Markov chain with stationary law Π using a transition kernel $K(x, \cdot)$ whose transitions can be written in a *stochastic recursive sequence* (SRS) form,

$$X_{t+1} = \phi(X_t, \xi_{t+1}), \quad t = 0, 1, 2, \ldots, \tag{8.1}$$

where ϕ is a deterministic map and ξ_{t+1} is a random variable with state space $\Lambda \subset \mathbb{R}^r$. (Sometimes it is automatically assumed that $\Lambda = (0,1)^r$, but that is not necessary here.) More precisely, the distribution of ξ is such that $P(X_{t+1} \in A) = \Pi(A) = \int K(x_t, A)\Pi(dx_t)$, that is, it guarantees that the output X_{t+1} has the same (marginal) distribution as the input X_t if $X_t \sim \Pi$.

To explain the key idea of CFTP, let us first review the usual implementation of MCMC. When the chain can be written as in Equation 8.1, we can simply compute it iteratively starting from an *arbitrary* starting point $X_0 \in \mathcal{S}$, by generating a sequence of $\xi_1, \xi_2, \ldots, \xi_t$, if we decide to run for t iterations. If the Markov chain formed by Equation 8.1 is positive recurrent and aperiodic (see Chapter 10, this volume), then we know that as $t \to \infty$, the probability law of X_t, P_t, will approach Π, regardless of the distribution of X_0. Of course, how large t needs to be before the difference between P_t and Π becomes too small to have practical consequences is the very thorny issue we try to avoid here.

The CFTP, as its name suggests, resolves this problem using an ingenious idea of running the chain *from the past* instead of *into the future*. To see this clearly, compare the following two sequences based on the same random sequence $\{\xi_1, \xi_2, \ldots, \xi_t\}$ used above:

$$\begin{aligned} \text{Forward: } X^{(x)}_{0\to t} &= \phi(\phi(\ldots\phi(\phi(x,\xi_1),\xi_2),\ldots\xi_{t-1}),\xi_t); \\ \text{Backward: } X^{(x)}_{t\to 0} &= \phi(\phi(\ldots\phi(\phi(x,\xi_t),\xi_{t-1}),\ldots\xi_2),\xi_1). \end{aligned} \tag{8.2}$$

The *time-forward* sequence $X^{(x)}_{0\to t}$ is obviously identical to the X_t computed previously with $X_0 = x$. The *time-backward* sequence $X^{(x)}_{t\to 0}$ is evidently not the same as $X^{(x)}_{0\to t}$ but clearly they have *identical distribution* whenever $\{\xi_1, \xi_2, \ldots, \xi_t\}$ are exchangeable, which certainly is the case when $\{\xi_t,\ t = 1, 2, \ldots\}$ are i.i.d., as in a typical implementation. (Note a slight abuse of notation: the use of t both as the length of the chain and as a generic index.) Consequently, we see that if we can somehow compute $X^{(x)}_{t\to 0}$ at its limit at $t = \infty$, then it will be a genuine draw from the desired distribution because it has the same distribution as $X^{(x)}_{0\to t}$ at $t = \infty$.

8.2.2 Hitting the Limit

At first sight, we seem to have accomplished absolutely nothing by constructing the time-backward sequence because computing $X^{(x)}_{t\to 0}$ at $t = \infty$ surely should be as impossible as computing $X^{(x)}_{0\to t}$ at $t = \infty$! However, a simple example reveals where the magic lies. Consider a special case where $\phi(X_t, \xi_{t+1}) = \xi_{t+1}$, that is, the original $\{X_t,\ t = 1, 2, \ldots\}$ already forms an i.i.d. sequence, which clearly has the distribution of ξ_1 as its stationary distribution (again, we assume $\{\xi_t,\ t = 1, 2, \ldots\}$ are i.i.d.). It is easy to see that in such cases, $X^{(x)}_{0\to t} = \xi_t$, but $X^{(x)}_{t\to 0} = \xi_1$, for all t. Therefore, with $X^{(x)}_{0\to\infty}$ we can only say that it has the *same distribution* as ξ_1, whereas for $X^{(x)}_{\infty\to 0}$ we can say *it is* ξ_1!

More generally, under regularity conditions, one can show that there exists a *stopping time* T such that $P(T < \infty) = 1$ and that the distribution of $X^{(x)}_{T\to 0}$ is exactly Π, that is, $X^{(x)}_{T\to 0}$ "hits the limit" with probability 1. Intuitively, this is possible because unlike $X^{(x)}_t \equiv X^{(x)}_{0\to t}$, which forms a Markov chain, $\tilde{X}^{(x)}_t \equiv X^{(x)}_{t\to 0}$ depends on the entire history of $\{\tilde{X}_1, \ldots, \tilde{X}_{t-1}\}$. It is this dependence that restricts the set of possible paths $\tilde{X}_t$ can take and hence makes it possible to "hit the limit" in a finite number of steps. For a mathematical proof of the existence of such T, we refer readers to [48,53,54].

The CFTP strategy, in a nutshell, is to identify the aforementioned stopping time T via coupling. To see how it works, let us first map t to $-t$ and hence relabel $X^{(x)}_{T\to 0}$ as $X^{(x)}_{-T\to 0}$, which makes the meaning *from the past* clearer. That is, there is a chain coming from the infinite past (and hence negative time) whose value at the present time $t = 0$ is the draw from the desired stationary distribution. This is because coming from infinite past and reaching the present time is mathematically the same as starting from the present time and reaching the infinite future. However, this equivalence will remain just a mathematical statement if we really have to go into the infinite past in order to determine the current value of the chain. But the fact that the backward sequence can hit the limit in a finite number of steps suggests that, for a given infinite sequence $\{\xi_t,\ t = -1, \ldots\}$, there exists a finite T such that, when $t \geq T$, $X^{(x)}_{-t\to 0}$ will no longer depend on x, that is, all paths determined by $\{\xi_t,\ t = -1, \ldots\}$ will coalesce by time 0, regardless of their origins at the infinite past. It was proved in [10] that such coupling is possible if and only if the Markov chain $\{X_1, X_2, \ldots\}$ determined by ϕ is uniformly ergodic.

8.2.3 Challenges for Routine Applications

Clearly once all paths coalesce, their common value $X_0 = X^{(x)}_{-T\to 0}$ is a genuine draw from the stationary law Π. Therefore, the CFTP protocol relies on our ability to design the MCMC process given by ϕ, or more generally by the transition kernel K, such that the *coalescence of all paths* takes place for moderate values of T. This requirement poses immediate challenges in its routine applications, especially for Bayesian computation, where $\mathcal{S}$ typically contains many states, very often uncountably many. The brute-force way of monitoring each path is infeasible for two reasons. First, it is simply impossible to follow infinitely many paths individually. Second, when the state space is continuous, even if we manage to reduce the process to just two paths (as with the monotone coupling discussed below), the probability that these will meet is zero if they are left to run independently. Therefore, our first challenge is to design the algorithm so that the number of paths shrinks to a finite one within a few steps. A hidden obstacle in this challenge is being able to figure out exactly *which* paths will emerge from this reduction process as they are the ones that need to be monitored until coalescence. The second challenge is to find effective ways to "force" paths to meet, that is, to couple them in such a way that, at each step, the probability that they take the same value is positive.

The rest of this chapter will illustrate a variety of methods designed to address both challenges and other implementation issues. We do not know any universal method, nor do we believe it exists. But there are methods for certain classes of problems, and some of them are rather ingenious.

8.3 Coalescence Assessment

8.3.1 Illustrating Monotone Coupling

Suppose that the space $\mathcal{S}$ is endowed with a partial order relationship $\prec$ so that

$$x \prec y \Rightarrow \phi(x, \xi) \leq \phi(y, \xi) \tag{8.3}$$

for any $x, y \in \mathcal{S}, \xi \in \Lambda$, and where ϕ is an SRS as in Equation 8.1. If we can find the minimum and maximum states $X_{\min}, X_{\max} \in \mathcal{S}$ with respect to the order $\prec$, then we can implement this *monotone coupler*—as defined by Equation 8.3—in which it is sufficient to verify the coupling of the paths started at these two extremal points because all other states are "squeezed" between them. Therefore, the monotone coupler is an efficient way to address the first challenge discussed in Section 1.2.3. For illustration, consider the random walk with state space $\mathcal{S} = \{0.25, 0.5, 2, 4\}$, with probability p of moving up or staying if the chain is already at the ceiling state $X_t = 4$, and probability $1 - p$ of moving down or staying if already at the floor state $X_t = 0.25$. It is easy to see that this construction forms a monotone chain, expressible as $X_t = \phi(X_{t-1}, \xi_t)$, where $\xi_t \sim \text{Bernoulli}(p)$ and its value determines the direction of the walk, with one going up and zero going down.

Figure 8.1 shows a realization of the CFTP process, corresponding to

$$\{\xi_{-8}, \xi_{-7}, \ldots, \xi_{-2}, \xi_{-1}\} = \{0, 1, 0, 1, 1, 1, 1, 0\}. \tag{8.4}$$

One can see that the order between paths is preserved by ϕ. In particular, all the paths are at all times between the paths started at $X_{\min} = 0.25$ (solid line) and $X_{\max} = 4$ (dashed line), respectively. Therefore, in order to check the coalescence of all four paths, we only need to check if the top chain starting from $X = 4$ and the bottom chain starting from $X = 0.25$ have coalesced. In this toy example, the saving from checking two instead of all four is obviously insignificant, but one can easily imagine the potentially tremendous computational savings when there are many states, such as with the Ising model applications in [48].

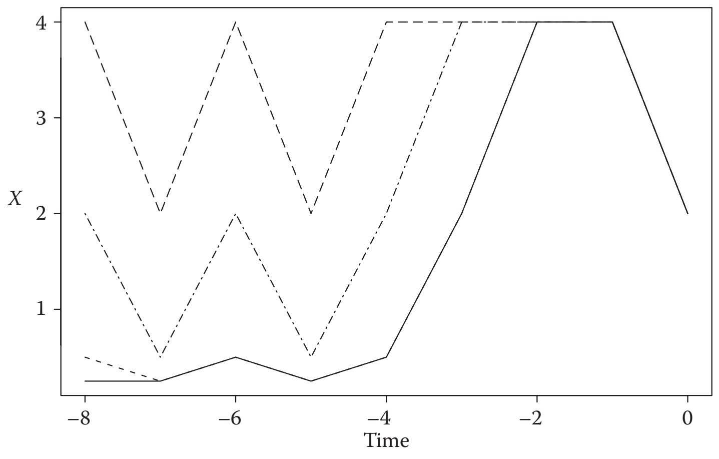

FIGURE 8.1
Illustration of a monotone SRS which preserves the natural order on the real line (i.e. paths can coalesce but never cross each other). Different lines represent sample paths started from different states.

8.3.2 Illustrating Brute-Force Coupling

This toy example also illustrates well the “brute-force” implementation of CFTP, that is, checking directly the coalescence of all paths. Figure 8.1 establishes that for any infinite binary sequences $\{\xi_t,\ t \le -1\}$, as long its last eight values (i.e. from $t = -8$ to $t = -1$) are the same as that given in Equation 8.4, the backward sequence given in Equation 8.2 will hit the limit $X = 2$, that is, the value of the coalesced chain at $t = 0$. Pretending that the monotone property was not noticed, we can still check the coalescence step by step for all paths. Or, more efficiently, we can use the “binary back-off” scheme proposed in [48]; that is, whenever a check fails to detect coalescence, we double the number of “backward” steps. Specifically, imagine we first made one draw of ξ, and it is zero (corresponding to $\xi_{-1} = 0$). We compute $X^{(x)}_{-1\to 0}$ of Equation 8.2 for all values of $x \in \mathcal{S}$, which leads to

$$X^{(4)}_{-1\to 0} = 2, \quad X^{(2)}_{-1\to 0} = 0.5, \quad X^{(0.5)}_{-1\to 0} = X^{(0.25)}_{-1\to 0} = 0.25,$$

indicating that coalescence has not occurred. We therefore double the number of steps going back, which requires only one new draw from $\xi \sim \text{Bernuolli}(p)$, as we already have $\xi_{-1} = 0$. It is important to emphasize here that we always *reuse* the draws of ξ_t that we have already made because the point here is to simply check what the coalesced value would be for a *given* infinite sequence of $\{\xi_t,\ t \le -1\}$. The device of making draws starting from $t = -1$ and going backward is the ingenious part of CFTP because it allows us to determine the property of an *infinite* sequence by revealing and examining only a *finite* number of its last elements. That is, since the remaining numbers in the (infinite) sequence cannot alter the value of the chain at $t = 0$, we do not even need to care what they are.

Now this new draw yields $\xi = 1$, and hence we have $\{\xi_{-2}, \xi_{-1}\} = \{1, 0\}$, which is then used to compute Equation 8.2 again but with $T = -2$:

$$X^{(4)}_{-2\to 0} = X^{(2)}_{-2\to 0} = 2, \quad X^{(0.5)}_{-2\to 0} = 0.5, \quad X^{(0.25)}_{-2\to 0} = 0.25,$$

hence, again, no coalescence. Once again we double the steps and go further back to $T = -4$, which means we need two more draws of ξ, and this time they both are one, yielding $\{\xi_{-4}, \xi_{-3}, \xi_{-2}, \xi_{-1}\} = \{1, 1, 1, 0\}$. Since we only need at most three consecutive upward steps to bring any state to the ceiling state $X = 4$, the $\{1, 1, 1, 0\}$ sequence immediately implies that

$$X^{(x)}_{-4\to 0} = \phi(4, 0) = 2, \quad \text{for all } x \in \mathcal{S}.$$

We have therefore detected coalescence after going back to only $T = -4$. This is not in any contradiction to Figure 8.1, but points to an even stronger statement that only the last four elements in the sequence (Equation 8.4), $\{\xi_{-4},\ \xi_{-3},\ \xi_{-2},\ \xi_{-1}\} = \{1, 1, 1, 0\}$, rather than all eight elements, are really relevant.

8.3.3 General Classes of Monotone Coupling

One may wonder when such ordering exists in more general situations and, if so, what important classes of distributions can be identified to satisfy (Equation 8.3). Such questions have been investigated by [13,18] in the case of *monotone* (also called *attractive*) and *anti-monotone* (also called *repulsive*) distributions Π. Suppose that $\mathcal{S} = \mathcal{Z}^d$, for some set $\mathcal{Z} \subset \mathbb{R}$.

We consider the componentwise partial order on $\mathcal{S}$ so that $x \prec y$ if and only if $x_i \leq y_i$ for all $1 \leq i \leq d$. The probability measure P on $\mathcal{S}$ is defined to be *monotone* if, for each $1 \leq i \leq d$,

$$P(X_i \leq s | X_{[-i]} = a) \geq P(X_i \leq s | X_{[-i]} = b), \quad \text{for all } s \in \mathcal{S}, \tag{8.5}$$

whenever $a \prec b$ in $\mathcal{Z}^{d-1}$, where $X_{[-i]} = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_d)$. Similarly, P is called *anti-monotone* if

$$P(X_i \leq s | X_{[-i]} = a) \leq P(X_i \leq s | X_{[-i]} = b),$$

whenever $a \prec b$ in $\mathcal{Z}^{d-1}$.

This definition of monotonicity via all full conditional distributions $P(X_i | X_{[-i]})$, $i = 1, \ldots, d$, was motivated by their use with the Gibbs sampler. In particular, Equations 8.3 and 8.5 are easily connected when the sampling from $P(X_i \leq s | X_{[-i]} = a)$ is done via the inverse CDF method. Put $F_i(s|a) = P(X_i \leq s | X_{[-i]} = a)$ and assume that the ith component is updated using $\phi_i(x, U) = (x_1, x_2, \ldots, x_{i-1}, \inf\{s : F_i(s|x_{[-i]}) = U\}, x_{i+1}, \ldots, x_d)$, with $U \sim U(0, 1)$. If we assume $x \prec y$, then from Equation 8.5 we get

$$\phi_i(x, U) \prec \phi_i(y, U) \tag{8.6}$$

because $\inf\{s : F_i(s|x_{[-i]}) = U\} \leq \inf\{s : F_i(s|y_{[-i]}) = U\}$. Applying Equation 8.6 in sequential order from $i = 1$ to $i = d$, as in a Gibbs sampler fashion, we can conclude that for $\vec{U} = \{U_1, \ldots, U_d\}$, the composite map

$$\phi(x, \vec{U}) = \phi_d(\phi_{d-1}(\ldots \phi_2(\phi_1(x, U_1), U_2), \ldots, U_{d-1}), U_d) \tag{8.7}$$

is monotone in x with respect to the same partial order $\prec$.

In the case of anti-monotone target distributions, it is not hard to see that the $\phi(x, \vec{U})$ of Equation 8.7 is also anti-monotone with respect to $\prec$ if d is odd, but monotone if d is even. Indeed, the ceiling/upper and floor/lower chains switch at each step (indexed by $i = 1$ to $i = d$), that is, the ceiling chain becomes the floor chain and vice versa. This oscillation behavior, however, still permits us to construct *bounding chains* that squeeze in between all the sample paths such that the general coalescence can be detected once the bounding chains have coalesced. See, for example, [13], which also discusses other examples of monotone target distributions; see also [6,21].

8.3.4 Bounding Chains

In a more general setup, [18] discusses the use of bounding chains without any condition of monotonicity. To better fix ideas, consider the following simple random walk with state space $\mathcal{S} = \{0.25, 0.5, 2\}$ and with transition probability matrix

$$A = \begin{pmatrix} p & 1-p & 0 \\ 0 & p & 1-p \\ p & 0 & 1-p \end{pmatrix},$$

where the $(1, 1)$th entry corresponds to the probability that the chain stays at 0.25. Unlike the previous random walk, the recursion defined by the matrix A is neither monotone nor anti-monotone with respect to the natural order on the real line. For example, with

$\xi \sim \text{Bernoulli}(p)$, and if $\xi = 1$, we have $\phi(0.25, \xi) = 0.25 < \phi(0.5, \xi) = 0.5 > \phi(2, \xi) = 0.25$, where ϕ is the chain's SRS. In contrast to the previous random walk, here $\xi = 1$ can indicate both moving up or down depending on the starting position, and this is exactly what destroys monotonicity with respect to the same ordering as in the previous random-walk example. (This, of course, by no means implies that no (partial) ordering existed under which the SRS is monotone; seeking such an ordering is indeed a common implementation strategy for perfect sampling.)

In Figure 8.2 we show one run of the CFTP algorithm implemented for this simple example with $p = 0.1$, where $\{\xi_{-8}, \ldots, \xi_{-1}\} = \{0, 1, 0, 0, 0, 0, 0, -1\}$. One can see that the three paths cross multiple times and no single path remains above or below all the others at all times. A bounding chain, in the general definition introduced by [18], is a chain $\{Y_t : t \geq 0\}$ defined on $2^{\mathcal{S}}$, the set of all subsets of $\mathcal{S}$, with the property that if $X_t^{(x)} \in Y_t$ for all $x \in \mathcal{S}$ then $X_{t+1}^{(x)} \in Y_{t+1}$ for all $x \in \mathcal{S}$; evidently Y_0 needs to contain all values in S. If, at some time t, Y_t is a singleton then coalescence has occurred. Clearly, there are many ways to define the chain Y_t, but only a few are actually useful in practice and these are obtained, usually, from a careful study of the original chain X_t.

For instance, in our example we notice that after one iteration $Y_0 = \mathcal{S}$ will become either $Y_1 = \{0.25, 0.5\}$ or $Y_1 = \{0.5, 2\}$, depending on whether $\xi = 1$ or $\xi = 0$, and therefore Y_t will always be a subset of these two sets (possibly themselves). Therefore, for $t \geq 1$, the updating rule $Y_{t+1} = \Psi(Y_t, \xi)$ can be simplified to

$$\Psi(Y_t, \xi) = \begin{cases} Y_t, & \text{if } \xi = 1 \text{ and } Y_t = \{0.25, 0.5\}, \\ \{0.25, 0.5\} & \text{if } \xi = 1 \text{ and } Y_t = \{0.5, 2\}, \\ \{0.5, 2\}, & \text{if } \xi = 0 \text{ and } Y_t = \{0.25, 0.5\}, \\ \{2\}, & \text{if } \xi = 0 \text{ and } Y_t = \{0.5, 2\}, \\ \phi(X_t, \xi), & \text{if } Y_t = \{X_t\}. \end{cases} \tag{8.8}$$

One can see then that having the ordered triplet $\{1, 0, 0\}$ in the ξ-sequence triggers coalescence, after which one simply follows the path to time zero.

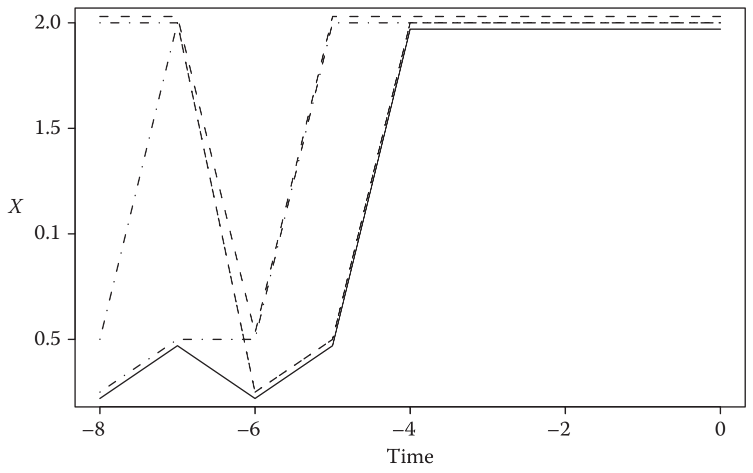

FIGURE 8.2
Nonmonotone Markov chain. The dashed and solid lines mark the bounding processes.

Two essential requirements for an effective bounding chain are that (i) it can detect coalescence of the original chain and (ii) it requires less effort than running all original sample paths. The chain $Y_t \equiv \{\mathcal{S}\}$ for all t is a bounding chain and satisfies (ii), but clearly it is useless. As an example of bounding chains that do not satisfy (ii), consider the dashed path and solid path in Figure 8.2. Here the dashed path is the maximum value attained by all paths at each time t, and the solid path is the minimal value (both have been slightly shifted for better visualization). For each time t, the interval between the dashed and the solid paths, denoted by $\tilde{Y}_t$, clearly forms a bounding chain. But unlike Y_t in Equation 8.8, the updating function for $\tilde{Y}_t$ is not easy to define, so running $\tilde{Y}_t$ involves checking the extremes of all the paths for X_t and is thus as complicated as running all paths for X_t.

As far as general strategies go, [13] shows how to construct bounding chains when each component of the random vector X is updated via a Gibbs sampler step, whereas [18] presents a general method for constructing bounding chains and applies it to problems from statistical mechanics and graph theory.

8.4 Cost-Saving Strategies for Implementing Perfect Sampling

The plain vanilla CFTP described in Section 8.2 suffers from two main drawbacks. First, the implementation “from the past” requires the random seeds used in the backward process to be stored until coupling is observed and a random sample is obtained. Second, the impatient user cannot abandon runs that are too long without introducing sampling bias, because the coupling time T is correlated with the sample obtained at time zero. In the following two sections we provide intuitive explanations of the read-once CFTP and Fill’s interruptible algorithm, designed respectively to address these two drawbacks.

8.4.1 Read-Once CFTP

Read-once CFTP (Ro-CFTP), as proposed by Wilson [56], is a clever device that turns CFTP into an equivalent “forward-moving” implementation. It collects the desired i.i.d. draws as the process moves forward and without ever needing to save any of the random numbers previously used. The method starts with a choice of a fixed block size K, such that the K-composite map

$$\phi_K(x;\vec{\xi}) = \phi(\phi(\ldots\phi(\phi(x,\xi_1),\xi_2),\ldots,\xi_{K-1}),\xi_K),$$

where $\vec{\xi} = \{\xi_1,\ldots,\xi_K\}$, has a high probability of coalescing, that is, the value of $\phi_K(x;\vec{\xi})$ will be free of x, or equivalently, all paths coalesce *within the block* defined by $\vec{\xi}$. In [56], Wilson suggests selecting K such that the probability of ϕ_K coalescing, denoted by p_K, is at least 50%. Given such a ϕ_K, we first initialize the process by generating i.i.d. $\vec{\xi}_j, j = 1, 2, \ldots,$ until we find a $\vec{\xi}_{j_0}$ such that $\phi_K(x;\vec{\xi}_{j_0})$ coalesces. Without loss of generality, in the top panel of Figure 8.3, we assumed that $j_0 = 1$; and we let $S_0 = \phi_K(x;\vec{\xi}_{j_0})$. We then repeat the same process, that is, generating i.i.d. $\vec{\xi}_j$s until $\phi_K(x;\vec{\xi})$ coalesces again. In the top panel of Figure 8.3, this occurred after three blocks. We denote the coalescent value as S_1. During this process, we follow from block to block only the *coalescence path* that goes through S_0 while all the other paths are reinitialized at the beginning of each block. The location of the coalescence path *just before* the beginning of the next coalescent composite map is a sample from the desired

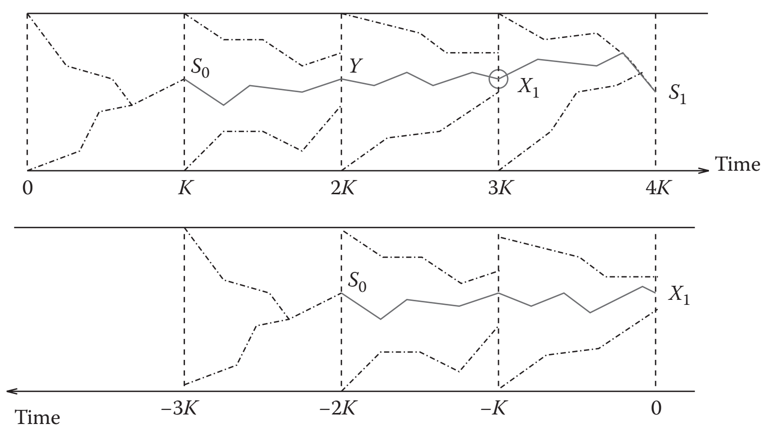

FIGURE 8.3
Top: The read-once CFTP with blocks of fixed length. Bottom: Comparison with CFTP2.

Π. In Figure 8.3 this implies that we retain the circled X_1 as a sample. The process then is repeated as we move forward, and this time we follow the path starting from S_1 and the next sample X_2 (not shown) is the output of this path immediately before the beginning of the next coalescent block. We continue this process to obtain i.i.d. draws X_3, X_4, and so on.

The connection between the Ro-CFTP and CFTP may not be immediately clear. Indeed, in the plain vanilla CFTP, the concept of a composite/block map is not emphasized because, although we "back off" in blocks, we do not require to have a coalescent composite map of fixed length. For instance, if we set $K = 4$, we can see that in Figure 8.1 the paths started at $-2K$ coalesce in the interval $(-K, 0)$ rather than within the block $(-2K, -K)$. However, suppose we consider a modified implementation of the plain vanilla CFTP, call it CFTP2, in which we go back from time zero block by block, each with size K, until we find a block that is coalescent, that is, all paths coalesce *within that block*. Clearly, if we trace the path from the coalescent value from that block until it reaches time zero, it will be exactly the same value as found by the original plain vanilla CFTP because once the coalescence takes place, all paths will stay together forever. The bottom panel of Figure 8.3 illustrates CFTP2, where the third block (counting backward from $t = 0$) is the coalescent block, and X_1 is our draw.

The resemblance of the bottom panel and the first three blocks in the top panel (counting forward from time $t = 0$) is intended to highlight the equivalence between Ro-CFTP and CFTP2. On its own, CFTP2 is clearly less cost-effective than CFTP because by insisting on having block coalescence, it typically requires going back further in time than does the original CFTP (since block coalescence is a more stringent detecting criterion, as discussed above). However, by sacrificing a little of the efficiency of detecting coalescence, we gain the *independence* between the block coalescent value S_0 and the entire backward search process for S_0, and hence we can reverse the order of the search without affecting the end result.

As this independence is the backbone of the Ro-CFTP, here we show how critically it depends on having *fixed-size* blocks. Intuitively, when the blocks all have the same size, they each have the same probability of being a coalescent block, and the distribution of the coalesced state given a coalescent block is the same regardless of which block it is. To confirm this intuition and see how it implies the independence, let us define the block random vector $\vec{\xi}_{-t} = (\xi_{-tK}, \xi_{-tK+1} \ldots, \xi_{-tK+K-1})$ and, for a given set $\vec{\xi}_{-t}, t = 1, 2, \ldots,$ let T

be the first t such that $\phi_K(x;\vec{\xi}_{-t})$ coalesces, and let $S_0 = \phi_K(x;\vec{\xi}_{-T})$ be the coalescent value. Also let $C_j = \{\phi_K(x,\vec{\xi}_{-j})$ coalesces$\}$, that is, the event that the jth block map coalesces. Then $\{T = t\} = (\cap_{j=1}^{t-1} C_j^c) \cap C_t$. For notational simplicity, denote $A_j = \{\phi_K(x,\vec{\xi}_{-j}) \in A\}$ and $B_j = \{\Xi_j \in B\}$, where A and B are two arbitrary (measurable) sets on the appropriate probability spaces, and $\Xi_j = \{\vec{\xi}_{-1},\ldots,\vec{\xi}_{-j}\}$. Then for any positive integer t,

$$\begin{aligned} P(S_0 \in A, T = t, \Xi_{T-1} \in B) &= P(A_t \cap [\cap_{j=1}^{t-1} C_j^c \cap C_t] \cap B_{t-1}) \\ &= P(A_t \cap C_t) P(\cap_{j=1}^{t-1} C_j^c \cap B_{t-1}) \\ &= P(A_t|C_t) P(C_t) P(\cap_{j=1}^{t-1} C_j^c \cap B_{t-1}) \\ &= P(A_t|C_t) P(C_t \cap_{j=1}^{t-1} C_j^c \cap B_{t-1}) \\ &= P(A_1|C_1) P(T = t, B_{T-1}). \end{aligned} \tag{8.9}$$

In deriving the above equalities, we have repeatedly used the fact that $\{A_t, C_t\}$ are independent of $\{A_{t-1}, B_{t-1}, C_{t-1}\}$ since they are determined respectively by $\vec{\xi}_{-t}$ and $\{\vec{\xi}_j,\ j = -1,\ldots,-(t-1)\}$. The last switching from $P(A_t|C_t)$ to $P(A_1|C_1)$ is due to the i.i.d. nature of the $\{A_t, C_t\}$, because all blocks have the same size K. This switching is critical in establishing the factorization in Equation 8.9, and hence the independence.

Clearly, as depicted in Figure 8.3, the output of CFTP2, namely X_1, can be expressed as $M(S_0, T, \Xi_{T-1})$, where M is a *deterministic* map. The aforementioned independence ensures that if we can find $\{\tilde{T}, \Xi_{\tilde{T}-1}\}$ such that it has the same distribution as $\{T, \Xi_{T-1}\}$ and is independent of S_0, then $\tilde{X}_1 = M(S_0, \tilde{T}, \Xi_{\tilde{T}-1})$ will have the same distribution as $X_1 = M(S_0, T, \Xi_{T-1})$, and hence it is also an exact draw from the stationary distribution Π. Because $\{\vec{\xi}_{-1}, \vec{\xi}_{-2},\ldots,\}$ are i.i.d., obviously the distribution of $\{T, \Xi_{T-1}\}$ is invariant to the order at which we check for the block coalescence. We can therefore reverse the original backward order into a forward one and start at an arbitrary block which must be independent of S_0. This naturally leads to the Ro-CFTP, because we can start with the block immediately after a coalescence has occurred (which serves as S_0), since it is independent of S_0. Moreover, the number of blocks and all the block random numbers (i.e. ξs) needed *before* we reach the next coalescent block represents a sample from the distribution of $\{T, \Xi_{T-1}\}$. (It is worth emphasizing that each coalescent composite map fulfills two roles as it marks the end of a successful run (inclusive) and the beginning of a new run (exclusive).) Alternatively, Equation 8.9 implies that we can first generate T from a geometric distribution with mean $1/p_K$ (recall that p_K is the probability of coalescence within each block), and then generate $T-1$ noncoalescent blocks, via which we then run the chain forward starting from S_0. This observation has little practical impact since p_K is usually unknown, but it is useful for understanding the connection with the splitting chain technique that will be discussed in Section 8.5. The forward implementation brought by Ro-CFTP also makes it easier to implement the efficient use of perfect sampling tours proposed by [40], which will be discussed in Section 8.6.

8.4.2 Fill's Algorithm

Fill's algorithm [8] and its extension to general chains [9] break the dependence between the backward time to coalescence and the sample obtained at time zero. In the following we use the slightly modified description from [39].

The algorithm relies on the time-reversal version of the Markov chain designed to sample from Π. If the original chain has transition kernel $K(x,\cdot)$, then the time-reversal version has kernel $\tilde{K}(z,\cdot)$, such that

$$\tilde{k}(x|z)\pi(z) = k(z \mid x)\pi(x), \quad \forall\,(x,z) \in \mathcal{S} \times \mathcal{S}, \tag{8.10}$$

where, for simplicity of presentation, we have assumed that the stationary law Π has density π, and $K(x,\cdot)$ and $\tilde{K}(z,\cdot)$ have kernel densities $k(\cdot|x)$ and $\tilde{k}(\cdot \mid z)$, respectively. It also requires that, given a particular path $X_0 \to X_1 \to \cdots \to X_t$, we can sample, *conditional on the observed path*, a sample of the same length from any state in $\mathcal{S}$.

The algorithm starts by sampling a random $Z \in \mathcal{S}$ from an arbitrary distribution P_0 (with density p_0) that is absolutely continuous with respect to Π, and by selecting a positive integer T. The first stage is illustrated in the top panel of Figure 8.4: using the reversal time chain, we simulate the path $Z = X_T \to X_{T-1} \to \cdots \to X_1 \to X_0$ (note that the arrow is pointing against the direction of time). In the second stage, we sample forward from all the states in $\mathcal{S}$ conditional on the existing path $X_0 \to X_1 \to \cdots \to X_T = Z$ (note that this path is considered now in the same direction as time). If by time T all the paths have

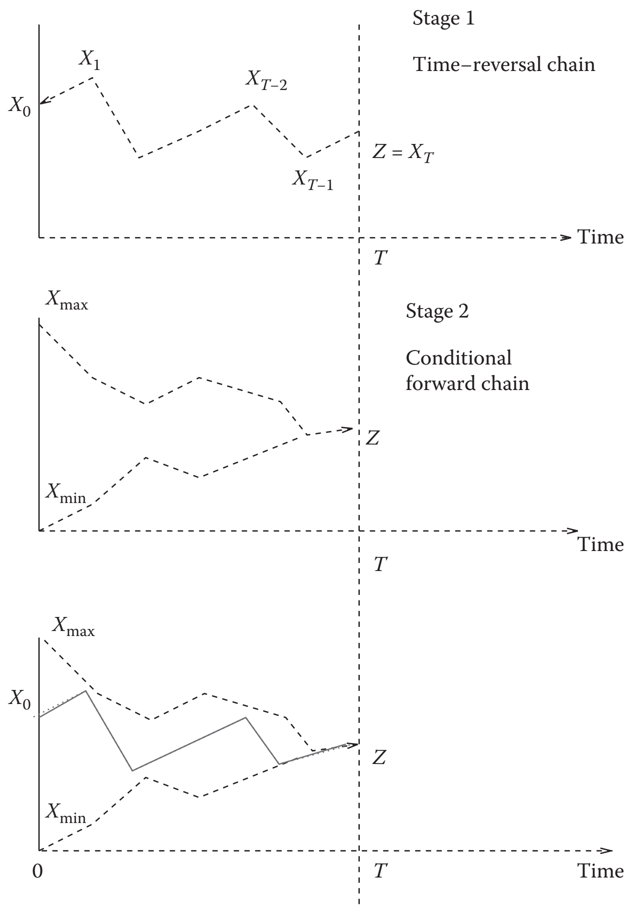

FIGURE 8.4
Illustration of Fill's algorithm.

coalesced, as depicted in the middle panel of Figure 8.4 (where we used monotone coupling for simplicity of illustration, but the idea is general), we retain X_0 as a sample from π, as shown in the bottom panel of Figure 8.4, and restart with a new pair (Z, T). Otherwise, we select a new T or we restart with a new pair (Z, T).

To understand why the algorithm produces i.i.d. samples from π, we first note that Equation 8.10 holds in the more general form

$$\tilde{k}_t(x|z)\pi(z) = k_t(z\,|\,x)\pi(x), \quad \forall\ (x, z) \in \mathcal{S} \times \mathcal{S}, \tag{8.11}$$

where k_t is the kernel density of the t-step forward transition kernel K_t and $\tilde{k}_t$ is the corresponding time-reversal one, $\tilde{K}_t$. Fill's algorithm retains only those paths from Z to X_0 (obtained via $\tilde{K}_T$) such that the corresponding $k_T(z|x)$ is free of x—and hence it can be expressed as $h_T(z)$—due to coalescence; in this sense Fill's algorithm is a case of rejection sampling. Therefore, using Equation 8.11, the probability density for those retained X_0s is

$$p(x) = \int \tilde{k}_T(x\,|\,z)p_0(z)\,dz = \int \frac{\pi(x)h_T(z)}{\pi(z)}p_0(z)\,dz \propto \pi(x), \tag{8.12}$$

hence the correctness of the sampling algorithm (see also [4]). Note that here, for simplicity, we have deliberately blurred the distinction between the fixed t in Equation 8.11 and the potentially random T in Equation 8.12; in this sense Equation 8.12 is a heuristic argument for building intuition rather than a rigorous mathematical proof. In its general form, Fill's algorithm can search for the coalescence time T just as with CFTP—see [9] for a detailed treatment of the general form of Fill's algorithm, including a rigorous proof of its validity. See also [4] for an alternative proof based directly on the rejection-sampling argument, as well as for a numerical illustration.

The conditional sampling is the main difficulty encountered when implementing Fill's algorithm, but in some cases it can be straightforward. For instance, if $X_{t+1} = \phi(X_t, U_{t+1})$ then it is possible that U_{t+1} is uniquely determined once both X_t and X_{t+1} are fixed (e.g. if we generate X_{t+1} using the inverse CDF transformation). If we denote by $\{U_1, \ldots, U_T\} := \mathcal{U}^0$ the set of random deviates determined by the path $Z = X_T \to X_{T-1} \to \cdots \to X_1 \to X_0$, then at the second stage we simply run for T steps, from all the states of $\mathcal{S}$, the Markov chains using the recursive form (Equation 8.1) with the set $\mathcal{U}^0$ as random seeds.

8.5 Coupling Methods

All algorithms described so far require the coupling of a finite or infinite number of paths in finite time. This is the greatest difficulty of applying perfect sampling algorithms to continuous state spaces, especially those with unbounded spaces (which is the case for most routine applications in Bayesian computation) and this is where the greatest ingenuity is required to run perfect sampling in more realistic settings. A good coupling method must be usable in practice and it is even better if it is implementable for different models with the same degree of success. In this section, we review some of the most useful coupling techniques, which essentially belong to two different types: (i) those which induce a "common regeneration state" that all sample paths must enter with positive probability; and (ii) those which explore hidden discretization and hence effectively convert the problem into one with a finite state space.

8.5.1 Splitting Technique

A very common technique for coupling MCMC paths is initiated in [46] and discussed in detail by [54]. Consider the Markov chain $\boldsymbol{X}_t$ defined using the transition kernel K and suppose that there exist $t > 0$, $0 < \epsilon < 1$, a set C (called a *small set*) and a *probability measure* ν such that

$$K_t(x, dy) \geq \epsilon\nu(dy), \quad \forall\, x \in C,$$

where K_t represents the t-step transition kernel. Thus, for any $x \in C$,

$$K_t(x, dy) = \epsilon\nu(dy) + (1-\epsilon)\frac{K_t(x, dy) - \epsilon\nu(dy)}{1-\epsilon} = \epsilon\nu(dy) + (1-\epsilon)Q(x, dy), \qquad (8.13)$$

where $Q(x, dy) = [K_t(x, dy) - \epsilon\nu(dy)]/(1-\epsilon)$. The representation given by Equation 8.13 is important because with probability ϵ the updating of the chain will be done using the probability measure ν, that is, *independently* of the chain's current state. If at time t all the paths are in the set C and all the updates use the same random numbers ξ that lead to the transition into the ν component of Equation 8.13, then all paths will coalesce at time $t + 1$, even if there are uncountably many. However, for a set $C \subset \mathcal{S}$ it will be difficult, if not impossible, to determine whether it contains all paths at a given time. This problem is alleviated in the case of CFTP where the existence of successful coupling has been shown (see [10]) to be equivalent to the uniform ergodicity of the chain $\boldsymbol{X}_t$, in which case the small set is the whole sample space, $\mathcal{S}$, so all paths are automatically within a small set at all times. An example where this idea has been brought to fruition is the *multigamma coupler* introduced by [37], following the gamma coupler of [25]. The method is further developed by [36] in the context of perfect sampling from continuous state distributions.

The multigamma coupler applies when the update kernel density $f(\cdot|x)$ of the Markov chain is known. In addition, it requires that there is a nonnegative function r such that

$$f(y \mid x) \geq r(y), \quad \forall x, y \in \mathcal{S}. \qquad (8.14)$$

If we denote $\rho = \int r(y)dy > 0$, then in line with the splitting technique discussed above we can write

$$P(X_{t+1} \leq y \mid X_t = x) = \rho R(y) + (1-\rho)Q(y|x), \qquad (8.15)$$

where $R(y) = \rho^{-1}\int_{-\infty}^{y} r(v)\,dv$ and $Q(y \mid x) = (1-\rho)^{-1}\int_{-\infty}^{y}[f(v \mid x) - r(v)]\,dv$.

As a simple example, assume that the transition kernel has the gamma density $f(y \mid a, b_x) = y^{a-1}b_x^a \exp(-yb_x)/\Gamma(a)$, where a is fixed, and b_x depends on the previous state $X_t = x$ but is always within a fixed interval, say $b_x \in [b_0, b_1]$, where b_0 and b_1 are known constants. Then we can set $r(y) = y^{a-1}b_0^a \exp(-yb_1)/\Gamma(a)$, which yields $\rho = (b_0/b_1)^a$. At each t, we sample $\xi \sim \text{Bernoulli}(\rho)$, and if $\xi = 1$, we draw y from Gamma(a, b_1), and let all paths $X_{t+1} = y$ regardless of their previous states, hence coalescence takes place. If $\xi = 0$, then we draw from the Q component in Equation 8.15 (though this step requires drawing from a nonstandard distribution).

In situations when no uniform bound can be found on $\mathcal{S}$ for Equation 8.14 to hold, Murdoch and Green [37] propose partitioning $\mathcal{S} = \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_m$ and bounding the kernel density f on each $\mathcal{S}_i$ with r_i and introduce a *partitioned multigamma coupler* for this setting. A more difficult coupling strategy has been described in [22] in the case of geometrically (but not necessarily uniformly) ergodic chains, though the approach has not been implemented on a wide scale.

There is a direct connection between the multigamma coupler and the Ro-CFTP in Section 8.4.1. With a block of size $K = 1$ the multigamma coupler construction implies that the probability of coalescence within the block is ρ. As described above, we can therefore sample a geometric T with success probability ρ, and start from a coalesced value, that is, an independent draw from $R(y)$ in Equation 8.15. We then run the chain forward for $T - 1$ steps conditioning on noncoalesced blocks, namely, we use the Q component of Equation 8.15 as the transition kernel. The resulting value then is an exact draw from Π [37].

There is also a close connection between the multigamma coupler and the slice sampler (see Section 8.5.4), as both can be viewed as building upon the following simple idea: For a given (not necessarily normalized) density $g(y)$, if (U, Y) is distributed uniformly on $\Omega_g = \{(u, y) : u \leq g(y)\}$, then the marginal density of Y is proportional to $g(y)$. Therefore, when $f(y|x) \geq r(y)$ for all x and y, we have

$$\Omega_r = \{(u, y) : u \leq r(y)\} \subset \Omega_{f,x} = \{(u, y) : u \leq f(y \mid x)\}, \quad \forall\, x \in \mathcal{S}. \tag{8.16}$$

For simplicity of illustration, let us assume that all $\Omega_{f,x}$ are contained in the unit square $[0, 1] \times [0, 1]$. Imagine now we use rejection sampling to achieve the uniform sampling on $\Omega_{f,x}$ for a particular x by drawing uniformly on the unit square. The chance that the draw (u, y) will fall into Ω_r is precisely ρ, and more importantly, if it is in Ω_r, it is an acceptable proposal for $f(y|x)$ regardless of the value of x because of Equation 8.16. This is the geometric interpretation of how the coalescence takes place for splitting coupling, which also hints at the more general idea of coupling via a common proposal, to which we now turn.

8.5.2 Coupling via a Common Proposal

The idea of using a common proposal to induce coalescence was given in [3] as a way to address the second challenge discussed in Section 1.2.3. (Note, however, that this strategy does not directly address the first challenge, namely discretizing a continuous set of paths into a finite set; that challenge is addressed by, for example, the augmentation method described in the next subsection, or by other clever methods such as the multishift coupler in [57].) Imagine that we have managed to reduce the number of paths to a finite one. In practice, it may still take a long time (possibly *too* long) before all paths coalesce into one. Intuitively, one would like to make it easier for paths that are close to each other to coalesce more quickly.

Remarkably, the description of coupling via a common proposal can be formulated in a general setting irrespective of the transition kernel used for the chain, as long as it has a density. Suppose that the chain of interest has transition kernel with the (conditional) density $f(\cdot|X_t)$. Instead of always accepting the next state as $X_{t+1} \sim f(\cdot|X_t)$, we occasionally replace it with a random draw $\tilde{Y}$ sampled from a user-defined density g. Thus, the X_{t+1} from the original chain plays the role of a proposal and is no longer guaranteed to be the next state; we therefore relabel it as $\tilde{X}_{t+1}$.

Instead, given $X_t = x$, the next state X_{t+1} is given by the updating rule

$$X_{t+1} = \begin{cases} \tilde{Y}, & \text{if } \dfrac{f(\tilde{Y}|x)g(\tilde{X}_{t+1})}{f(\tilde{X}_{t+1}|x)g(\tilde{Y})} > U, \\[2ex] \tilde{X}_{t+1}, & \text{otherwise,} \end{cases} \tag{8.17}$$

where $U \sim U(0,1)$ and is independent of any other variables. In other words, the above coupler makes a choice between two independent random variables $\tilde{X}_{t+1}$ and $\tilde{Y}$ using a Metropolis–Hastings (MH) acceptance ratio. Note that the MH accept–reject move is introduced here simply to ensure that the next state of the chain has distribution density $f(\cdot|X_t)$ even if occasionally the state is "proposed" from g. The coupling via a common proposal tends to increase the propensity of coalescing paths that are close to each other. More precisely, suppose that two of the paths are close, that is, $X_t^{(1)} \approx X_t^{(2)}$. Then the ratios (Equation 8.17) will tend to be similar for the two chains, which implies that both chains will likely accept/reject $\tilde{Y}$ simultaneously.

It is also worth emphasizing that the above scheme needs a modification in order to be applicable to the MH sampler which does not admit a density with respect to the Lebesgue measure. The basic idea is to introduce the common proposal into the MH proposals themselves as in [3]. This perhaps is best seen via a toy example. Suppose that our target distribution is $N(0,1)$, and we adopt a random-walk Metropolis algorithm, that is, the proposal distribution is $q(y|x) = N(y-x)$, where $N(z)$ is the density of $N(0,1)$. Clearly, because $N(z)$ is continuous, two paths started at different points in the sample space will have zero probability of coalescing if we just let them "walk randomly." To stimulate coalescence, we follow the ideas in [3] and create an intermediate step in which the proposals used in the two processes can be coupled.

More precisely, at each time t we sample $\tilde{Z}_{t+1} \sim t_3(\cdot)$, where t_3 is the t distribution with three degrees of freedom. Suppose that the proposal for chain i at time t is $\tilde{Y}_{t+1}^{(i)}$, where $\tilde{Y}_{t+1}^{(i)} \sim N(X_t^{(i)}, 1)$. We then define

$$\tilde{W}_{t+1}^{(i)} = \begin{cases} \tilde{Z}_{t+1}, & \text{if } \dfrac{N(\tilde{Z}_{t+1} - X_t^{(i)})t_3(\tilde{Y}_{t+1}^{(i)})}{N(\tilde{Y}_{t+1}^{(i)} - X_t^{(i)})t_3(\tilde{Z}_{t+1})} > U, \\[2ex] \tilde{Y}_{t+1}^{(i)}, & \text{otherwise,} \end{cases} \tag{8.18}$$

where $U \sim U(0,1)$ is independent of all the other variables. The proposal $\tilde{W}_{t+1}^{(i)}$ is accepted using the usual MH strategy because its density is still the density of the original proposal, $N(X_t^{(i)}, 1)$; the next state is then either $\tilde{W}_{t+1}^{(i)}$ (acceptance) or $X_t^{(i)}$ (rejection). What has changed is that regardless of which paths the chains have taken, their MH proposals now have a positive probability of taking on a common value $\tilde{Z}_{t+1}$ for all those chains for which the first inequality in Equation 8.18 is satisfied. This does not guarantee coupling, but it certainly makes it more likely. In Figure 8.5 we show two paths simulated using the simple model described above, where the two paths first came very close at $t = -8$ and then coalesced at $t = -7$.

8.5.3 Coupling via Discrete Data Augmentation

Data augmentation [51], also known in statistical physics as the auxiliary variable method, is a very effective method for constructing efficient MCMC algorithms; see [55] for a review. It turns out to be useful for perfect sampling as well, because we can purposely consider auxiliary variables that are discrete and therefore convenient for assessing coalescence. Specifically, suppose that our target density is $f(x)$, where x may be continuous. Suppose that we have a way to augment $f(x)$ into $f(x,l)$, where l is discrete. If we can perform Gibbs sampling via $f(x \mid l)$ and $f(l|x)$, then we will automatically have a Markov sub-chain with

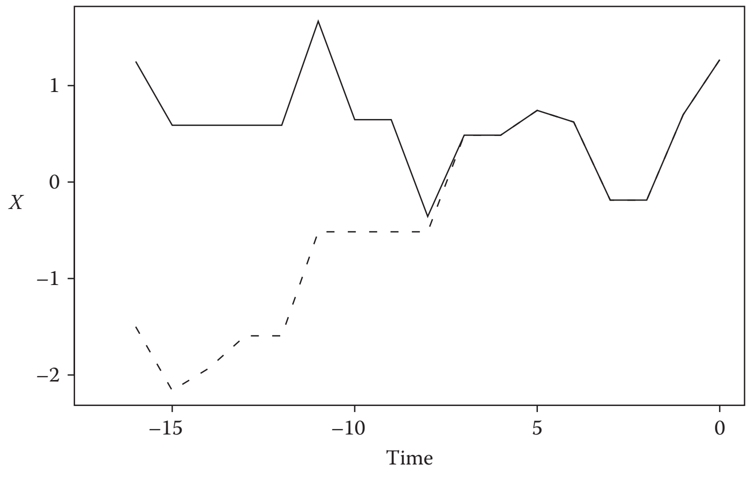

FIGURE 8.5
Illustration of coupling with proposals for two paths.

$f(l)$ as the stationary density (note the sub-chain with l only is Markovian because the Gibbs sampler here only involves two steps). Therefore, we have effectively turned the continuous problem for $f(x)$ into a discrete one because once we have an authentic draw from $f(l)$, then we can easily get a corresponding authentic draw from $f(x)$ by sampling from $f(x \mid l)$.

To illustrate, consider finite mixtures, where the obvious auxiliary variable is the indicator variable indicating the mixture component from which a sample is obtained. The coupling via augmentation has been successfully implemented by [17] in the case of two-component mixtures of distributions and by [38] in the case of Bayesian mixture priors. Below is one of the examples discussed by [17], which we recast in order to crystalize the essence of discrete data augmentation.

Consider the mixture $\alpha f_0(d) + (1-\alpha)f_1(d)$, where only the mixture proportion α is unknown and therefore we seek its posterior density, assuming a uniform prior on $(0, 1)$. Given a sample $\{d_1, \ldots, d_n\}$ from the mixture, the posterior for α is proportional to

$$p(\alpha|\vec{d}) \propto \prod_{i=1}^{n} \{\alpha f_0(d_i) + (1-\alpha)f_1(d_i)\}, \tag{8.19}$$

involving 2^n terms when expanded; note that here we use $\vec{d} = \{d_1, \ldots, d_n\}$ to denote the data instead of the original $\{x_1, \ldots, x_n\}$, to avoid the potential confusion of our generic notation which uses X for the sampling variable, which is α here. Let the latent variables $\vec{z} = \{z_1, \ldots, z_n\}$ be such that $z_i = 0$ if d_i has been generated from f_0 and $z_i = 1$ if d_i has been generated from f_1. Then it is easy to see that

$$P(z_i = 1|\vec{d}, \alpha) = \frac{(1-\alpha)f_1(d_i)}{\alpha f_0(d_i) + (1-\alpha)f_1(d_i)} := p_i \tag{8.20}$$

and

$$P(\alpha|\vec{z}) = \text{Beta}\left(n + 1 - \sum_{i=1}^{n} z_i, \sum_{i=1}^{n} z_i + 1\right). \tag{8.21}$$

This implies that we can construct the discrete augmentation as $l = \sum_i z_i$, which has a nonhomogenous binomial (NhB) distribution NhB$(n, \vec{p})$, where $\vec{p} = \{p_1, \ldots, p_n\}$. That is, l is the sum of n independent but not necessarily identically distributed Bernoulli variables. Given this data augmentation scheme $f(\alpha, l)$, the algorithm given in [17] can be reformulated as follows.

1. Because of Equation 8.21, given $l_t = l$, we generate $\alpha_{t+1} \sim \text{Beta}(n+1-l, l+1)$, which can be accomplished by drawing $w_j \sim \text{Exponential}(1)$ for $j \in \{1, \ldots, n+2\}$ and then letting

$$\alpha_{t+1} = \frac{\sum_{i=1}^{n+1-l} w_i}{\sum_{i=1}^{n+2} w_i}. \tag{8.22}$$

2. Given $\alpha_{t+1} = \alpha$, because of Equation 8.20, we need to draw l_{t+1} from NhB$(n, \vec{p}(\alpha))$, where $\vec{p}(\alpha) = \{p_1, \ldots, p_n\}$, with $p_i \equiv p_i(\alpha)$ given by the right-hand side of Equation 8.20. This draw is accomplished by generating independent $u_i \sim U(0, 1)$ and letting

$$l_{t+1} = \sum_{i=1}^{n} \mathbf{1}\{u_i \leq p_i\}, \tag{8.23}$$

where $\mathbf{1}\{A\}$ is the usual indicator function of event A.

Combining Equations 8.22 and 8.23, we see that the SRS from l_t to l_{t+1} can be written as

$$l_{t+1} \equiv \phi(l_t; \vec{u}, \vec{w}) = \sum_{i=1}^{n} \mathbf{1}\left\{u_i \leq \left[1 + \left(\frac{\sum_{i=1}^{n+2} w_i}{\sum_{i=1}^{n+1-l_t} w_i} - 1\right)^{-1} \frac{f_0(d_i)}{f_1(d_i)}\right]^{-1}\right\}. \tag{8.24}$$

For given $\vec{u} = \{u_1, \ldots, u_n\}$ and $\vec{w} = \{w_1, \ldots, w_n\}$, the function ϕ in Equation 8.24 is evidently increasing in l_t and thus defines, with respect to the natural integer ordering, a monotone Markov chain on the state space $S_l = \{0, \ldots, n\}$, with the ceiling and floor states given by $l = 0$ and $l = n$. Through data augmentation we have therefore converted the problem of drawing from the continuous distribution given by Equation 8.19 into one in which the sample space is the finite discrete space S_l, given by Equation 8.24, for which we only need to trace the two extreme paths starting from $l = 0$ and $l = n$.

8.5.4 Perfect Slice Sampling

Slice sampling is based on the simple observation that sampling from Π (assumed to have density π) is equivalent to sampling from the uniform distribution $g(u, x) \propto \mathbf{1}\{u \leq f(x)\}$, where f is the unnormalized version of π and is assumed known. One can easily see that the marginal distribution of x is then the desired one. In turn, the sampling from g can be performed using a Gibbs scan in which both steps involve sampling from uniform distributions:

I. Given X_t, sample $U \sim U(0, f(X_t))$.
II. Given U from Step I, sample $X_{t+1} \sim U[A(U)]$, where $A(w) = \{y : f(y) \geq w\}$.

Here, for simplicity, we assume that $A(U)$ has finite Lebesgue measure for any U; more general implementations of the slice sampler are discussed in [7,45]. The coupling for slice sampling has been designed by [30] under the assumption that there exists a minimal element $x_{\min} \in \mathcal{S}$ with respect to the order $x \prec y \Leftrightarrow f(x) \leq f(y)$.

Specifically, the *perfect slice sampler* achieves coupling via introducing common random numbers into the implementation of Steps I and II in the following fashion. We implement Step I, regardless of the value of X_t, by drawing $\epsilon \sim U(0,1)$ and then letting $U = U(X_t) = \epsilon f(X_t)$; hence all the $U(X_t)$ share the same random number ϵ.

Given the $U = U(X_t)$ from Step I, we need to implement Step II in such a way that there is a positive (and hopefully large) probability that all X_{t+1} will take the same value regardless of the value X_t. This is achieved by forming a sequence of random variables $\mathbf{W} = \{W_j\}_{j=1,2,\ldots}$, where $W_1 \sim U[A(f(x_{\min}))]$ and $W_j \sim U[A(f(W_{j-1}))]$, for any $j \geq 2$. The desired draw X_{t+1} is then the first $W_j \in A(U(X_t)) = A(\epsilon f(X_t))$, that is,

$$X_{t+1} \equiv \phi(X_t, (\epsilon, \mathbf{W})) = W_{\tau(X_t)},$$

where $\tau(x) = \inf\{j : f(W_j) \geq \epsilon f(x)\}$.

In [30] it is proven that, almost surely, only a finite number of the elements of the sequence $\mathbf{W}$ are needed in order to determine $\tau(x)$. The correctness of the algorithm is satisfied if $W_{\tau(x)} \sim U[A(\epsilon f(x))]$, and in [30] this is established by viewing it as a special case of adaptive rejection sampling. Here we provide a simple direct proof. For any given x, denote $A^{(x)} = A(\epsilon f(x))$ and $B_j^{(x)} = \{(W_1, \ldots, W_j) : f(W_i) < \epsilon f(x),\ i = 1, \ldots, j\}$. Then clearly, for any $k \geq 1$, $\{\tau(x) = k\} = \{W_k \in A^{(x)}\} \cap B_{k-1}^{(x)}$ (assume $B_0^{(x)} = \mathcal{S}$ for any $x \in \mathcal{S}$). Hence, for any (measurable) set $C \subset A^{(x)}$, we have

$$\begin{aligned} P(\{W_{\tau(x)} \in C\}|\tau(x) = k) &= \frac{P(\{W_k \in C \cap A^{(x)}\} \cap B_{k-1}^{(x)})}{P(\{W_k \in A^{(x)}\} \cap B_{k-1}^{(x)})} \\ &= \frac{\mathrm{E}\left[\mathrm{E}\left(\mathbf{1}\{W_k \in C\}\mathbf{1}\{B_{k-1}^{(x)}\}|W_1, \ldots, W_{k-1}\right)\right]}{\mathrm{E}\left[\mathrm{E}\left(\mathbf{1}\{W_k \in A^{(x)}\}\mathbf{1}\{B_{k-1}^{(x)}\}|W_1, \ldots, W_{k-1}\right)\right]} \\ &= \frac{\mathrm{E}\left[\mathbf{1}\{B_{k-1}^{(x)}\}P(\{W_k \in C\}|W_{k-1})\right]}{\mathrm{E}\left[\mathbf{1}\{B_{k-1}^{(x)}\}P(\{W_k \in A^{(x)}\}|W_{k-1})\right]}. \end{aligned} \tag{8.25}$$

In the above derivation, we have used the fact that $\{W_1, \ldots, W_k\}$ forms a Markov chain itself. Given $W_{k-1} = w$, W_k is uniform on $A(f(w))$ by construction, so $P(\{W_k \in B\}| W_{k-1} = w) = \mu(B)/\mu(A(f(w)))$, where μ is the Lebesgue measure. Consequently, the last ratio in Equation 8.25 is exactly $\mu(C)/\mu(A^{(x)})$, the uniform measure on $A^{(x)}$. It follows immediately that $W_{\tau(x)} \sim U(A^{(x)}) = U[A(\epsilon f(x))]$.

To visualize how Steps I and II achieve coupling, Figure 8.6 depicts the update for two paths in the simple case in which f is strictly decreasing with support $(0, x_{\min})$. Suppose that the two chains are currently in X_1 and X_2. Given the ϵ drawn in Step I, the monotonicity of f allows us to write $A(\epsilon f(X_1)) = (0, A_1)$ and $A(\epsilon f(X_2)) = (0, A_2)$. Step II then starts by sampling $W_1 \sim U(0, x_{\min})$ and, since it is not in either of the intervals $(0, A_1)$ or $(0, A_2)$, we follow by sampling uniformly $W_2 \sim U(0, W_1)$ which is the same as sampling $W_2 \sim U[A(f(W_1))]$ since f is decreasing. Because $W_2 \in (0, A_2)$, we have $\tau(X_2) = 2$ so X_2 is updated

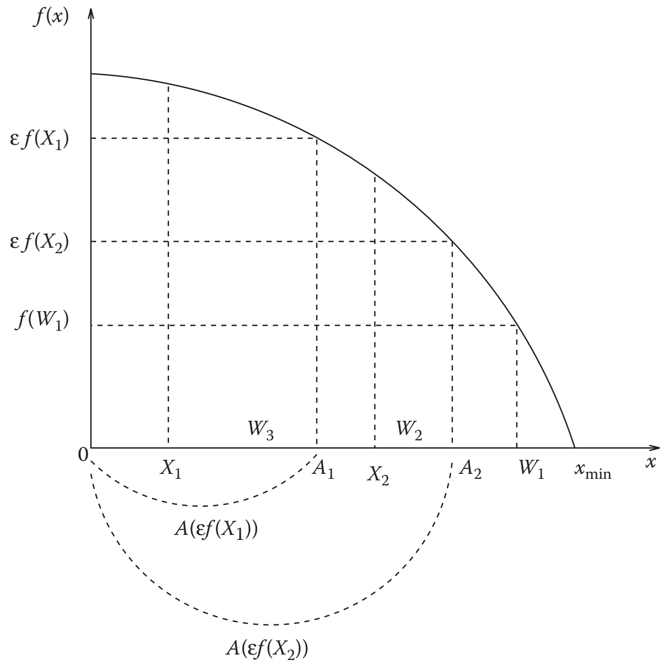

FIGURE 8.6
Illustration of perfect slice sampling.

into W_2. As $W_2 \notin (0, A_1)$ we continue by sampling $W_3 \sim U(0, W_2)$, and since $W_3 \in (0, A_1)$ we can set $\tau(X_1) = 3$. Thus, in the case illustrated by Figure 8.6, the updates are $\phi(X_1, \mathbf{W}) = W_3$ and $\phi(X_2, \mathbf{W}) = W_2$. To understand why this construction creates an opportunity for coupling, imagine that the second uniform draw, W_2, happens to be smaller than A_1. In this case, $\tau(X_1) = \tau(X_2) = 2$ so both X_1 and X_2 are updated into W_2, which means that the two paths have coalesced. In fact, for all $X \in (0, x_{\min})$ with the property that $f(X) \leq f(W_1)/\epsilon$ we have $\phi(X, \mathbf{W}) = W_1$, for all X such that $f(W_1)/\epsilon < f(X) \leq f(W_2)/\epsilon$ we have $\phi(X, \mathbf{W}) = W_2$, and so on. This shows how the continuous set of paths is discretized in only one update.

Figure 8.6 also illustrates that the density ordering $X_2 \prec X_1$ (since $f(X_2) < f(X_1)$) is consistent with the same ordering for the updates: $\phi(X_2, \mathbf{W}) = W_2 \prec \phi(X_1, \mathbf{W}) = W_3$ because $f(W_2) \leq f(W_3)$ by construction. This is true in general because if $X_2 \prec X_1$, that is, $f(X_2) \leq f(X_1)$, then $\tau(X_2) \leq \tau(X_1)$ because $A(\epsilon f(X_1)) \subset A(\epsilon f(X_2))$. Consequently, $W_{\tau(X_2)} \prec W_{\tau(X_1)}$. This property implies that we can implement the monotone CFTP as described in Section 1.3.1, if a maximal $x_{\max}$ exists. In situations in which the extremal states cannot be found, Mira et al. [30] show how to construct bounding processes for this perfect slice sampler.

8.6 Swindles

The term "swindle" has traditionally been used in the Monte Carlo literature to characterize any strategy or modification that either reduces the computational effort or increases the efficiency of the algorithm [12,50]. Usually, swindles are relatively easy-to-implement generic methods applicable to a wide class of algorithms. In the following we describe some of the swindles proposed that either are for or take advantage of perfect sampling algorithms.

8.6.1 Efficient Use of Exact Samples

The algorithms presented in Section 8.2 may be very slow in producing *one draw* from the distribution of interest Π, which is very uneconomical considering that the whole generation process involves a large number of random variables. Motivated by this observation, [40] proposed alternative implementations and estimators that extract more than one draw from each perfect sampling run.

One natural idea is that once a perfect draw is made, say $X_0 \sim \Pi$, then we obviously can run the chain forward for, say, k steps, all of which will be genuine draws from Π. However, this introduces serial correlation in the k samples retained for estimation. Simulations performed in [40] show that a more efficient implementation is the concatenated CFTP (CCFTP). The strategy is illustrated in Figure 8.7, in which two consecutive runs of the monotone CFTP have produced *independent* sample points $X, Y \sim \Pi$. Instead of using just the two sample points, CCFTP uses the *tour* made up of all the realizations lying on the path starting at $-T_Y$ (the time needed to detect coalescence for sampling Y) that connects X to Y, that is, the dashed line in Figure 8.7.

Since the time order of X and Y is irrelevant here (because all the random numbers are i.i.d. along the sequence), one could construct another tour using the path that starts at time $-T_X$ with state Y and ends in X; note that such a path must exist because all tours, regardless of their initial position, must coalesce after T_X iterations by design. Whereas such constructions are not hard to generalize to situations with more than two chains, it is much more straightforward to construct the tours with the Ro-CFTP algorithm. That is, in Figure 8.3, instead of using just X_1, we include in our sample the segment of the coalescence path between X_1 and X_2 (the second sample point not shown in the figure), and then between X_i and X_{i+1} for all i.

Clearly all such tours are independent, but the samples within one tour are serially correlated. If we denote the length of the ith tour by T_i and draws within the tour by X_{ij}, $j = 1, \ldots, T_i$, then obviously a consistent estimator for

$$I_g = \int g(x)\pi(dx) \tag{8.26}$$

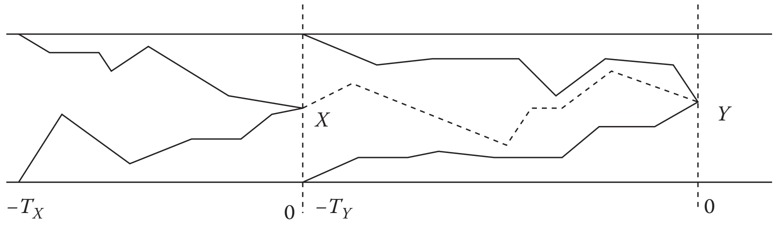

FIGURE 8.7
Illustration of a perfect tour.

that utilizes N tours is [44]

$$\hat{I}_g = \frac{\sum_{i=1}^{N}\sum_{j=1}^{T_i} g(X_{ij})}{\sum_{i=1}^{N} T_i} = \frac{\sum_{i=1}^{N} T_i \bar{g}_i}{\sum_{i=1}^{N} T_i},$$

where $\bar{g}_i$ is the sample average of $g(X_{ij})$ within the ith tour. Note, however, that this is a ratio estimator since the tour lengths are random, but the added benefit here is that since the tours are i.i.d., so are the $\{T_i, \bar{g}_i\}$. Hence, the variance of $\hat{I}_g$ can easily be estimated (when N is large) by the usual variance estimator for ratio estimators based on i.i.d. samples (see [24]), namely

$$\widehat{\text{var}}(\hat{I}_g) = \frac{\sum_{i=1}^{N} T_i{}^2(\bar{g}_i - \hat{I}_g)^2}{(\sum_{i=1}^{N} T_i)^2}. \tag{8.27}$$

Note that in Equation 8.27, for the sake of simplicity, we have used denominator N instead of the usual $N-1$ in defining the cluster sample variance—we can view each tour as a cluster of draws and there are N clusters in total. The beauty of Equation 8.27 is that it entirely avoids the issue of dealing with within-tour dependence, just as in the usual cluster sampling we do not need to be concerned with intra-cluster correlations when we have genuine replications of clusters, which are the tours here.

8.6.2 Multistage Perfect Sampling

Perfect sampling offers the possibility of performing simple random sampling. It is well known in the sampling survey literature that simple random sampling is surpassed in statistical and/or cost efficiency by a number of alternative sampling protocols, for instance multistage sampling. In light of this observation [27] proposed a different approach for running a monotone CFTP algorithm. The *multistage backward coupling algorithm* is designed to perform multistage sampling within the CFTP protocol.

For the sake of simplicity, we describe first the case with two stages. Consider a partition of the sample space into m clusters $\mathcal{S} = \cup_{i=1}^{m} \mathcal{C}_i$. In the first stage we run the CFTP until *cluster coalescence* occurs, that is, all chains merge into a common cluster, say $\mathcal{C}_j$ at time 0. In the second stage, we run CFTP to sample from the conditional distribution $\Pi(\cdot|\mathcal{C}_j)$ defined via $\Pi(A|\mathcal{C}_j) = \Pi(A)/\Pi(\mathcal{C}_j)$ for any measurable $A \subset \mathcal{C}_j$. The two-stage method can easily be extended to multiple stages using a class of nested partitions—for example, each element of the partition used in the second step can in turn be partitioned, $\mathcal{C}_j = \cup_{h=1}^{K_j} \mathcal{C}_{jh}$, and sampling from $\Pi(\cdot|\mathcal{C}_j)$ can be done again in two or more stages, and so on.

The astute reader may have realized that this procedure is valid only if the detection of cluster $\mathcal{C}_j$ in the first stage guarantees that the sample we would eventually have obtained was indeed going to belong to $\mathcal{C}_j$. One way to achieve this is to restrict the proposal to $\mathcal{C}_j$ when using MH algorithm for the second stage. In general, this "foretelling" requirement can be quite stringent when implemented in brute-force ways; more effective methods need to be developed before the multistage sampling methods see general applications. Nevertheless, when the method can be implemented, empirical evidence provided in [27] demonstrates that substantial reductions (e.g. 70%) in running time are possible.

8.6.3 Antithetic Perfect Sampling

Stratified sampling is another efficient method widely used in sample surveys. An implicit way of performing stratification in Monte Carlo simulations can be implemented via *antithetic variates* [15]. Traditionally, antithetic sampling is performed in Monte Carlo using two negatively correlated copies of an unbiased estimator. Suppose that we are interested in estimating Equation 8.26. Traditional Monte Carlo uses an i.i.d. sample $\{X_1, \ldots, X_{2n}\}$ from Π and the estimator

$$\hat{I}_{2n} = \frac{\sum_{i=1}^{2n} g(X_i)}{2n}.$$

The variance of $\hat{I}_{2n}$ can be significantly decreased if we are able to sample, for each $1 \le j \le n$, $X_j^{(1)}, X_j^{(2)} \sim \Pi$ such that $\mathrm{corr}(g(X_j^{(1)}), g(X_j^{(2)})) < 0$ and use the estimator

$$\tilde{I}_{2n} = \frac{\sum_{i=1}^{n} \left[g(X_i^{(1)}) + g(X_i^{(2)}) \right]}{2n}.$$

In the case where g is monotone, there are relatively simple ways to generate the desired pairs $(X_j^{(1)}, X_j^{(2)})$. Moreover, we have shown in [6] that increasing the number of *simultaneous* negatively correlated samples can bring a significant additional variance reduction. The more complex problem of generating $k \ge 3$ random variables $\{X^{(1)}, \ldots, X^{(k)}\}$ such that any two satisfy $\mathrm{corr}(g(X^{(i)}), g(X^{(j)})) \le 0$ can be solved, at least for monotone g, using the concept of *negative association* (NA) introduced in [20]. The random variables $\{X_i,\ i = 1, \ldots, k\}$, where each X_i can be of arbitrary dimension, are said to be negatively associated (NA) if for every pair of disjoint finite subsets A_1, A_2 of $\{1, \ldots, k\}$ and for any nondecreasing functions g_1, g_2,

$$\mathrm{cov}(g_1(X_i, i \in A_1), g_2(X_j, j \in A_2)) \le 0,$$

whenever the above covariance function is well defined. In light of this stringent condition it is perhaps not surprising that NA is a stronger form of negative dependence which is preserved by concatenation. More precisely, if $\{X_1, \ldots, X_{k_1}\}$ and $\{Y_1, \ldots, Y_{k_2}\}$ are two independent sets of NA random variables, then their union, $\{X_1, \ldots, X_{k_1}, Y_1, \ldots, Y_{k_2}\}$, is also a set of NA random variables. A number of methods used to generate vectors of NA random deviates, especially the very promising iterative Latin hypercube sampling, are discussed in [6].

The implementation of the antithetic principle for CFTP is relatively straightforward. Given a method to generate NA $\{\xi^{(1)}, \ldots, \xi^{(k)}\}$ (where ξ is as needed in Equation 8.1), one can run k CFTP processes in parallel, the jth one using $\{\xi_t^{(j)},\ t \le 0\}$, where $\{\xi_t^{(1)}, \ldots, \xi_t^{(k)}\}$, $t \le 0$, are i.i.d. copies of $\{\xi^{(1)}, \ldots, \xi^{(k)}\}$, as sketched in Figure 8.8. Within the jth process of CFTP all paths are positively coupled because they use the same $\{\xi_t^{(j)},\ t \le 0\}$. At each update, $\{\xi_t^{(1)}, \ldots, \xi_t^{(k)}\}$ are NA, a property that clearly does not alter the validity of each individual CFTP process.

To obtain $n = km$ draws, we repeat the above procedure *independently* m times, and collect $\{X_i^{(j)},\ 1 \le i \le m;\ 1 \le j \le k\}$, where i indexes the replication, as our sample $\{X_1, \ldots, X_n\}$. Let

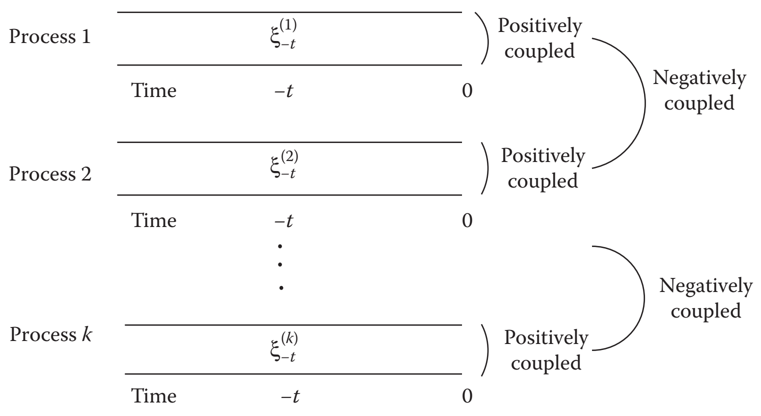

FIGURE 8.8
Parallel antithetic backward CFTP processes.

$\sigma_g^2 = \text{var}[g(X)]$ and $\rho_k^{(g)} = \text{corr}(g(X_1^{(1)}), g(X_1^{(2)}))$. Then

$$\text{var}\left(\frac{1}{n}\sum_{i=1}^{n} g(X_i)\right) = \frac{\sigma_g^2}{n}\left[1 + (k-1)\rho_k^{(g)}\right].$$

Consequently, the *variance reduction factor* (VRF), relative to the independent sampling *with the same simulation size*, is

$$S_k^{(g)} = 1 + (k-1)\rho_k^{(g)}.$$

We emphasize here the dependence of $S_k^{(g)}$ on k, and more importantly on g, and thus the actual gain in reduction can be of practical importance for some g but not for others, but $S_k^{(g)} \le 1$ as long as $\rho_k^{(g)} \le 0$.

8.6.4 Integrating Exact and Approximate MCMC Algorithms

It is probably clear by now to the statistician with some travel experience in the MCMC kingdom that perfect sampling may not be the vehicle that one could take on every trip. But it is possible to extend its range considerably if we couple it with more traditional MCMC methods. Here we describe such an approach devised by [33] to deal with Bayesian computation in cases where the sampling density is known only up to a constant that depends on the model parameter, and hence the likelihood function itself cannot be evaluated directly.

More precisely, consider the case in which the target of interest is the posterior density $\pi(\theta|y) \propto p(\theta)p(y\,|\,\theta)$, where $p(\theta)$ is the prior density and $p(y\,|\,\theta)$ is the sampling density of the data. There is a large spectrum of problems (e.g. Markov random fields, image analysis, Markov point processes, Gaussian graphical models, neural networks) for which $p(y\,|\,\theta)$ is known only up to a constant, that is, $p(y\,|\,\theta) = q(y\,|\,\theta)/C_\theta$, with the functional form of q known but the normalizing constant C_θ unknown, in the sense that its value at any particular θ is hard or even impossible to calculate. Obviously, for such problems, the classical MCMC approach cannot be directly implemented. For instance, a Metropolis algorithm with a symmetric proposal, moving from $\theta \to \theta'$, would require the calculation of the acceptance ratio

$$\alpha(\theta';\theta) = \min\left\{1, \frac{p(\theta')q(y\,|\,\theta')}{p(\theta)q(y\,|\,\theta)} \times \frac{C_\theta}{C_{\theta'}}\right\}$$

which involves the unknown ratio of two normalizing constants, $C_\theta/C_{\theta'}$, a problem which occurs in many areas (see, e.g., [11,28,29]).

One obvious way to deal with this problem is to use Monte Carlo or other approximations to estimate each ratio needed in the implementation of Metropolis–Hastings algorithm. A more creative and "exact" solution is proposed by [33] with the help of perfect sampling. The idea is to add into the mix an auxiliary variable x such that the chain updates not only θ but (θ, x) via MH sampling with an acceptance ratio in which no unknown constant appears. Since the auxiliary variable is just a computational artifact, as long as the marginal distribution of θ is preserved there is a lot of freedom in choosing how to update x. In particular, we consider updating (θ, x) via a proposal (θ', x') in which the proposal θ' is generated as in the original chain (it does not depend on x) but $x'|\theta', \theta, x \sim q(\cdot|\theta')/C_{\theta'}$. Essentially, x' is pseudo-data simulated from the sampling distribution when the parameter is equal to the proposal, θ'. For the new chain, the acceptance ratio is then

$$\tilde{\alpha} = \min\left\{1, \frac{p(\theta')q(y|\theta')q(x|\theta)}{p(\theta)q(y|\theta)q(x'|\theta')}\right\}, \tag{8.28}$$

which no longer involves any unknown normalizing constant.

The perceptive reader may immediately have realized that the above scheme simply transfers one difficult problem into another, namely, simulating from the original sampling density $p(\cdot|\theta') = q(\cdot|\theta')/C_{\theta'}$. Since $C_{\theta'}$ is not available, direct methods such as inverse CDF are out of the question (even when they are applicable otherwise). We can of course apply the Metropolis–Hastings algorithm itself for this sampling, which will not require any value of C_θ (since here we sample for x, not θ). But then we would need to introduce a new proposal, and more critically we would need to worry about the convergence of this imbedded Metropolis–Hastings algorithm *within each step* of creating a proposal (θ', x') as called for by Equation 8.28. This is clearly cumbersome and, indeed, entirely defeats the purpose of introducing x' in order to have a "clean" solution to the problem without invoking any approximation (beyond the original Metropolis–Hastings algorithm for θ). This is where the perfect sampling methodologies kick in, because if we have an exact draw from $p(x'|\theta')$, then the acceptance ratio given in Equation 8.28 is exactly correct for implementing the Metropolis–Hastings algorithm for drawing (θ, x) and hence for θ. This is particularly fitting, since intractable likelihoods are common in inference for point processes and this is also the area where exact sampling has been most successful. For instance, in [33], the method is illustrated on the well-known Ising model which was proposed as a main application in Propp and Wilson's landmark paper [48], which is a "must" for any tourist of the magic land of perfect sampling.

In closing, we should mention that the method discussed here is only one among a number of promising attempts that have been made to couple the power of traditional MCMC to the precision of perfect sampling such as in [40,43]. See also [42] for related ideas and algorithms.

8.7 Where Are the Applications?

The most resounding successes of perfect sampling have been reported from applications involving finite state spaces, especially in statistical physics (e.g. [13,19,48,49]) and point processes (e.g. [1,14,21,23,26,31,34,35,52]). Other applications include sampling from truncated distributions (e.g. [2,47]), queuing ([41]), Bayesian inference (as in [16,32,37,38]), and mixture of distributions (see [5,17]).

Whereas applications are many, and some are exceedingly successful, much still needs to be done before perfect sampling can be applied routinely. What is gained by perfect sampling is its "perfectness," that is, once it delivers a draw, we are theoretically guaranteed that its distribution is mathematically the same as our desired distribution. The price one pays for this mathematical precision is that any perfect sampling method refuses to produce a draw unless it is absolutely perfect, much like a craftsman reputed for his fixation with perfection refuses to sell a product unless it is 100% flawless. In contrast, any "nonperfect" MCMC method can sell plenty of its "products," but it will either ask the consumers to blindly trust their qualities or leave the consumers to determine their qualities at their own risk. The perfect sampling versus nonperfect sampling is therefore a tradeoff between quality and quantity. As with anything else in life, perhaps the future lies in finding a sensible balance. Perfect quality in small quantity only excites treasure collectors, and lousy quality in abundant quantity only helps garbage collectors. The future of perfect sampling methods lies in how successfully we can strike a balance—producing many quality products at an affordable price in terms of users' implementation cost.

Acknowledgments

We would like to thank Jeff Rosenthal for helpful discussions and three exceptionally thorough anonymous referees for comments and suggestions. RVC has been partially supported by the Natural Sciences and Engineering Research Council of Canada and XLM partially by the National Science Foundation of USA.

References

1. Berthelsen, K. K. and Møller, J. 2002. A primer on perfect simulation for spatial point processes. *Bulletin of the Brazilian Mathematical Society (N.S.)*, 33(3):351–367.
2. Beskos, A. and Roberts, G. O. 2005. One-shot CFTP; application to a class of truncated Gaussian densities. *Methodology and Computing in Applied Probability*, 7(4):407–437.
3. Breyer, L. and Roberts, G. O. 2000. Some multi-step coupling constructions for Markov chains. Technical report, Lancaster University.
4. Casella, G., Lavine, M., and Robert, C. P. 2001. Explaining the perfect sampler. *American Statistician*, 55:299–305.
5. Casella, G., Mengersen, K. L., Robert, C. P., and Titterington, D. M. 2002. Perfect slice samplers for mixtures of distributions. *Journal of the Royal Statistical Society*, *Series B*, 64:777–790.

6. Craiu, R. V. and Meng, X. L. 2005. Multi-process parallel antithetic coupling for forward and backward MCMC. *Annals of Statistics*, 33(2):661–697.
7. Damien, P., Wakefield, J., and Walker, S. 1999. Gibbs sampling for Bayesian non-conjugate and hierarchical models by using auxiliary variables. *Journal of the Royal Statistical Society*, *Series B*, 61:331–344.
8. Fill, J. A. 1998. An interruptible algorithm for perfect sampling via Markov chains. *Journal of Applied Probability*, 8:131–162.
9. Fill, J. A., Machida, M., Murdoch, D. J., and Rosenthal, J. S. 1999. Extension of Fill's perfect rejection sampling algorithm to general chains. *Random Structures and Algorithms*, 17(3–4): 290–316.
10. Foss, S. G. and Tweedie, R. L. 1998. Perfect simulation and backward coupling. *Stochastic Models*, 14:187–203.
11. Gelman, A. and Meng, X.-L. 1998. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185.
12. Gentle, J. 1998. *Random Number Generation and Monte Carlo Methods*. Springer, New York.
13. Häggström, O. and Nelander, K. 1998. Exact sampling from anti-monotone systems. *Statistica Neerlandica*, 52:360–380.
14. Häggström, O., van Lieshout, M. N. M., and Møller, J. 1999. Characterization results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes. *Bernoulli*, 5(4):641–658.
15. Hammersley, D. C. and Morton, K. V. 1956. A new Monte Carlo technique: antithetic variates. *Mathematical Proceedings of the Cambridge Philosophical Society*, 52:449–475.
16. Haran, M. 2003. Efficient perfect and MCMC sampling methods for Bayesian spatial and components of variance models. PhD thesis, Department of Statistics, University of Minnesota.
17. Hobert, J. P., Robert, C. P., and Titterington, D. M. 1999. On perfect simulation for some mixtures of distributions. *Statistics and Computing*, 9:287–298.
18. Huber, M. 2004. Perfect sampling using bounding chains. *Annals of Applied Probability*, 14(2):734–753.
19. Huber, M. L. 2002. A bounding chain for Swendsen-Wang. *Random Structures and Algorithms*, 22:43–59.
20. Joag-Dev, K. and Proschan, F. 1983. Negative association of random variables with applications. *Annals of Statistics*, 11:286–295.
21. Kendall, W. S. 1998. Perfect simulation for the area-interaction point process. In L. Accardi and C. C. Heyde (eds), *Probability Towards 2000*, pp. 218–234. Springer, New York.
22. Kendall, W. S. 2004. Geometric ergodicity and perfect simulation. *Electronic Communications in Probability*, 9:140–151.
23. Kendall, W. S. and Møller, J. 2000. Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Advances in Applied Probability*, 32(3):844–865.
24. Kish, L. 1965. *Survey Sampling*. Wiley, New York, 1965.
25. Lindvall, T. 1992. *Lectures on the Coupling Method*. Wiley, New York.
26. McKeague, I. W. and Loizeaux, M. 2002. Perfect sampling for point process cluster modelling. In A. B. Lawson and D. G. T. Denison (eds), *Spatial Cluster Modelling*, pp. 87–107. Chapman & Hall/CRC, Boca Raton, FL.
27. Meng, X.-L. 2000. Towards a more general Propp-Wilson algorithm: Multistage backward coupling. In N. Madras (ed.), *Monte Carlo Methods*, pp. 85–93. American Mathematical Society, Providence, RI.
28. Meng, X.-L. and Shilling, S. 2002. Warp bridge sampling. *Journal of Computational and Graphical Statistics*, 11(3):552–586.
29. Meng, X.-L. and Wong, W. 1996. Simulating ratios of normalizing constants via a simple identity: A theoretical exploration. *Statistica Sinica*, 6:831–860.
30. Mira, A., Møller, J., and Roberts, G. O. 2001. Perfect slice samplers. *Journal of the Royal Statistical Society*, *Series B*, 63(3):593–606.

31. Møller, J. 1999. Markov chain Monte Carlo and spatial point processes. In *Stochastic Geometry (Toulouse, 1996)*, pp. 141–172. Chapman & Hall/CRC, Boca Raton, FL.
32. Møller, J. and Nicholls, G. K. 1999. Perfect simulation for sample-based inference. Preprint, Department of Mathematical Statistics, Chalmers Institute of Technology.
33. Møller, J., Pettitt, A. N., Reeves, R., and Berthelsen, K. K. 2006. An efficient Markov chain Monte Carlo method for distributions with intractable normalising constants. *Biometrika*, 93(2):451–458.
34. Møller, J. and Rasmussen, J. G. 2005. Perfect simulation of Hawkes processes. *Advances in Applied Probability*, 37(3):629–646.
35. Møller, J. and Waagepetersen, R. P. 2004. *Statistical Inference and Simulation for Spatial Point Processes*. Chapman & Hall/CRC, Boca Raton, FL.
36. Murdoch, D. 2000. Exact sampling for Bayesian inference: Unbounded state spaces. In N. Madras (ed.), *Monte Carlo Methods*, pp. 111–121. American Mathematical Society, Providence, RI.
37. Murdoch, D.J. and Green, P. J. 1998. Exact sampling from a continuous state space. *Scand. J. Stat.*, 25:483–502.
38. Murdoch, D.J. and Meng, X.-L. 2001. Towards perfect sampling for Bayesian mixture priors. In E. George (ed.), *Proceedings of ISBA 2000*, pp. 381–390. Eurostat.
39. Murdoch, D.J. and Rosenthal, J. S. 1998. An extension of Fill's exact sampling algorithm to non-monotone chains. Technical report, University of Toronto.
40. Murdoch, D.J. and Rosenthal, J. S. 2000. Efficient use of exact samples. *Statistics and Computing*, 10:237–243.
41. Murdoch, D.J. and Takahara. G. 2006. Perfect sampling for queues and network models. *ACM Transactions on Modeling and Computer Simulation*, 16:76–92.
42. Murray, I. 2005. Advances in Markov chain Monte Carlo methods. PhD thesis, University College London.
43. Murray, I., Ghahramani, Z., and MacKay, D. 2006. MCMC for doubly-intractable distributions. In *Proceedings of the 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 359–366. AUAI Press, Corvallis, OR.
44. Mykland, P., Tierney, L., and Yu, B. 1995. Regeneration in Markov chain samplers. *Journal of the American Statistical Association*, 25:483–502.
45. Neal, R. M. 2003. Slice sampling (with discussion). *Annals of Statistics*, 31(3):705–767.
46. Nummelin, E. 1978. A splitting technique for Harris recurrent Markov chains. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 43(4):309–318.
47. Philippe, A. and Robert, C. 2003. Perfect simulation of positive Gaussian distributions. *Statistics and Computing*, 13:179–186.
48. Propp, J. G. and Wilson, D. B. 1996. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1–2):223–252.
49. Servidea, J. D. and Meng, X.-L. 2006. Statistical physics and statistical computing: A critical link. In J. Fan and H. Koul (eds), *Frontiers in Statistics: Dedicated to Peter John Bickel in Honor of his 65th Birthday*, pp. 327–344. Imperial College Press, London.
50. Simon, G. 1976. Computer simulation swindles, with applications to estimates of location and dispersion. *Applied Statistics*, 25:266–274.
51. Tanner, M. A. 1996. *Tools for Statistical Inference : Methods for the Exploration of Posterior Distributions and Likelihood Functions*, 3rd edn. Springer, New York.
52. Thönnes, E. 1999. Perfect simulation of some point processes for the impatient user. *Advances in Applied Probability*, 31(1):69–87.
53. Thönnes, E. 2000. A primer on perfect simulation. In K. Mecke and D. Stoyan (eds), *Statistical Physics and Spatial Statistics. The Art of Analyzing and Modeling Spatial Structures and Pattern Formation*, Lecture Notes in Physics 554, pp. 349–378. Springer, Berlin.
54. Thorisson, H. 2000. *Coupling, Stationarity, and Regeneration*. Springer, New York.
55. van Dyk, D. and Meng, X.-L. 2001. The art of data augmentation (with discussion). *Journal of Computational and Graphical Statistics*, 10:1–111.

56. Wilson, D. B. 2000a. How to couple from the past using a read-once source of randomness. *Random Structures Algorithms*, 16(1):85–113.
57. Wilson, D. B. 2000b. Layered multishift coupling for use in perfect sampling algorithms (with a primer on CFTP). In N. Madras (ed.), *Monte Carlo Methods*, pp. 141–176. American Mathematical Society, Providence, RI.