

Pseudo-Perfect and Adaptive Variants of the Metropolis-Hasting Algorithm with an Independent Candidate Density

J. N. Corcoran* and U. Schneider†

*The University of Colorado and

†The National Center for Atmospheric Research

January 14, 2004

Abstract

We describe and examine an imperfect variant of a perfect sampling algorithm based on the Metropolis-Hastings algorithm that appears to perform better than a more traditional approach in terms of speed and accuracy. We then describe and examine an “adaptive” Metropolis-Hasting algorithm which generates and updates a self-target candidate density in such a way that there is no “wrong choice” for an initial candidate density. Simulation examples are provided.

1 Introduction

The Metropolis-Hastings algorithm [17] allows simulation of a probability density $\pi(x)$ (with respect to Lebesgue measure, say, when $\mathbb{X} = \mathbb{R}^k$) which is known only up to a factor: that is, when only $\pi(x)/\pi(y)$ is known.

In order to describe the Metropolis-Hastings algorithm(s), we consider a *candidate transition kernel* $Q(x, A)$ for $x \in \mathbb{R}^k$ and a Borel set $A \subset \mathbb{R}^k$ satisfying

$$Q(x, A) \geq 0 \quad \text{and} \quad Q(x, \mathbb{R}^k) = 1$$

which generates potential transitions for a discrete-time Markov chain evolving on \mathbb{R}^k . We assume that there exists a density $q(x, y)$ such that

$$Q(x, A) = \int_A q(x, y) dy.$$

*Postal Address: Department of Applied Mathematics, University of Colorado, Box 526 Boulder CO 80309-0526, USA; email: corcoran@colorado.edu

†Postal Address: Geophysical Statistics Project, National Center for Atmospheric Research, Climate and Global Dynamics Division, P.O. Box 3000, Boulder, CO 80307-3000, USA; email: uli@ucar.edu

⁰Keywords: invariant measures, backwards coupling, coupling from the past, exact sampling, perfect sampling, Metropolis-Hastings

AMS Subject classification: 60J27, 65C05, 65C40

The Metropolis algorithm [11], dating back to 1953, uses a symmetric candidate transition Q for which $q(x, y) = q(y, x)$. In 1970, Hastings [8] extended the Metropolis algorithm to a more general Q . In either case, the simulator proceeds by generating candidate transitions from state x to state y according to the distribution Q , and accepting the transition with probability

$$\alpha(x, y) = \begin{cases} \min \left\{ \frac{\pi(y) q(x, y)}{\pi(x) q(y, x)}, 1 \right\} & \pi(x) q(y, x) > 0 \\ 1 & \pi(x) q(y, x) = 0. \end{cases}$$

Thus evolves a Markov chain with transition density

$$p(x, y) = q(x, y) \alpha(x, y), \quad y \neq x,$$

which will remain at the same point with probability

$$P(x, \{x\}) = \int q(x, y) [1 - \alpha(x, y)] dy.$$

It is easy to verify that π is the *invariant* (or stationary) measure for the chain:

$$\pi(A) = \int \pi(x) P(x, A) dx, \quad \forall A \in \mathcal{B}(X)$$

where X is the state space of the chain and $\mathcal{B}(X)$ are the Borel sets in X .

In order to simulate (sample) a value drawn from π , one must generally select a distribution Q and run a Metropolis-Hastings sample path for “a long time” until it is suspected that convergence to π has been achieved. Choices for Q and rates of convergence have been studied extensively in [[1],[10],[15]], for example.

In [3], a Metropolis-Hastings based “perfect sampling” algorithm was introduced based on the backward coupling approach of Propp and Wilson [14], eliminating the need to address issues of convergence. A drawback of this algorithm, described in Section 3.2, is that it is necessary to be able to maximize a certain ratio of densities. In Section 3.3, we implement an “imperfect” perfect sampling algorithm by using only an approximate maximum in the algorithm of [3]. Empirical results suggest that it is worthwhile to use the perfect sampling algorithm even with this introduced error, as it appears to outperform the traditional non-backward coupling approach.

With a traditional Metropolis-Hastings algorithm, a perfect version, or even an imperfect perfect version, there remains the issue of choosing a suitable candidate distribution Q . Algorithms with self-targeting candidates exist [15], and in a similar spirit we introduce an “adaptive” Metropolis-Hastings algorithm in Section 4. Preliminary results suggest that this approach considerably speeds up convergence and, in fact, will converge when the traditional approach may fail altogether.

2 The Independent Metropolis-Hastings Algorithm

In this paper we consider the so-called “independent” Metropolis-Hastings (IMH) chain, where we have a given candidate distribution Q which we will assume to have a density q ,

positive everywhere for convenience, with which we can generate potential values of an i.i.d. sequence. (We use the term “independent” to describe the Metropolis-Hastings algorithm where candidate states are generated by a distribution that is independent of the current state of the chain.) A “candidate value”, y , generated according to Q is then accepted with probability $\alpha(x, y)$ given by

$$\alpha(x, y) = \begin{cases} \min \left\{ \frac{\pi(y) q(x)}{\pi(x) q(y)}, 1 \right\} & \pi(x)q(x) > 0 \\ 1 & \pi(x)q(x) = 0, \end{cases} \quad (1)$$

where x is the current state of the chain. Thus, actual transitions of the IMH chain take place according to a law P with transition density

$$p(x, y) = q(y)\alpha(x, y), \quad y \neq x$$

and with probability of remaining at the same point given by

$$P(x, \{x\}) = \int q(y)[1 - \alpha(x, y)] dy.$$

With this choice of α we have that π is invariant for P .

It is known [10] that the IMH chain has desirable convergence properties (and, for example, is uniformly ergodic) if there exists a $\beta > 0$ such that

$$\frac{q(x)}{\pi(x)} \geq \beta, \quad (2)$$

and we will call q *overdispersed* with respect to π if this holds.

3 The Effect of Random Maximization on a Perfect Metropolis-Hastings Algorithm

3.1 Perfect Simulation

There has been considerable recent work on the development and application of “perfect sampling” algorithms that will enable the simulation of the invariant (or stationary) measure π of a Markov chain, either exactly (that is, by drawing a random sample known to be from π) or approximately, but with computable order of accuracy. These were sparked by the seminal paper of [14], and several variations and extensions of this idea have appeared since – see [4, 5, 6, 7, 9, 12], and [13]. These ideas have proven effective in areas such as statistical physics, spatial point processes and operations research, where they provide simple and powerful alternatives to methods based on iterating transition laws, for example.

The essential idea of most of these approaches is to find a random epoch $-T$ in the past such that, if we construct sample paths (according to a transition law $P(x, y)$ that is invariant for π) from every point in the state space starting at $-T$, then all paths will have coupled successfully by time zero. The common value of the paths at time zero is a draw from π .

Intuitively, it is clear why this result holds with such a random time T . For consider a chain starting at $-\infty$ with the stationary distribution π . At every iteration it maintains the distribution π . But at time $-T$ it must pick *some* value x , and from then on it follows the trajectory from that value. But of course it arrives at the same place at time zero no matter what value x is picked at time $-T$: so the value returned by the algorithm at time zero must itself be a draw from π .

Perfect sampling algorithms can be particularly efficient if the chain is *stochastically monotone* in the sense that paths from lower starting points stay below paths from higher starting points. In this case, one need only couple sample paths from the “top” and “bottom” of the space, as all other paths will be sandwiched in between. It is possible to generalize one step further to monotone chains on an unbounded state space by considering *stochastically dominating* processes to bound the journeys of sample paths.

For this to be practicable we need to ensure that T is indeed finite. [14] show that this occurs for irreducible aperiodic finite space chains, and for a number of stochastically monotone chains possessing maximal and minimal elements. Indeed, [3] show that if the distribution at $-\infty$ is any fixed (or even random) value x_0 , then under fairly standard conditions the value at time zero is still distributed according to π , and this observation is crucial in what follows.

There are several easy-to-read perfect sampling tutorials available, and we refer interested readers to [2]. In this paper we only wish to emphasize that the key idea in the search successively further and further back in time for the so-called *backward coupling time* T requires that one reuse random number streams. That is, if sample paths run forward to time 0 from time -1 using a random number (or random vector) U_{-1} have not coalesced by time 0, then one must go back further, say to time -2 and run paths forward for two steps using a random number U_{-2} and then the **previously used** U_{-1} .

We now describe a specific perfect sampling scheme of Corcoran and Tweedie [3] based on the Metropolis-Hastings algorithm.

3.2 The IMH Algorithm

Certain monotonicity properties of the “independent” Metropolis-Hastings (IMH) scheme, which we reviewed briefly in Section 2, are such that perfect sampling is feasible [3].

The perfect IMH algorithm uses the ratios in the acceptance probabilities given by (1) to reorder the states in such a way that we always accept moves to the left (or downwards). That is, if we write $\pi(x) = kh(x)$ where k is unknown, we define the IMH ordering,

$$x \succeq y \quad \Leftrightarrow \quad \frac{\pi(y)q(x)}{\pi(x)q(y)} \geq 1 \quad \Leftrightarrow \quad \frac{h(y)}{q(y)} \geq \frac{h(x)}{q(x)} \quad (3)$$

With this ordering, we can (hopefully) attain a “lowest state” l . Essentially, one can think of l as the state that is hardest to move away from when running the IMH algorithm. Thus, if we are able to accept a move from l to a candidate state y drawn from the distribution

Q with density q , then sample paths from every point in the state space will also accept a move to y , so all possible sample paths will couple. The perfect sampling algorithm is formally described as follows:

IMH (Backward Coupling) Algorithm

1. Draw a sequence of random variables $Q_n \sim Q$ for $n = 0, -1, -2, \dots$, and a sequence $\alpha_n \sim \text{unif}(0, 1)$ for $n = -1, -2, \dots$
2. For each time $-n = -1, -2, \dots$, start a lower path L at l , and an upper path, U at Q_{-n} .
3. (a) For the lower path: Accept a move from l to Q_{-n+1} at time $-n + 1$ with probability $\alpha(l, Q_{-n+1})$, otherwise remain at state l . That is, accept the move from l to Q_{-n+1} if $\alpha_{-n} \leq \alpha(l, Q_{-n+1})$.
 (b) For the upper path: Similarly, accept a move from Q_{-n} to Q_{-n+1} at time $-n + 1$ if $\alpha_{-n} \leq \alpha(Q_{-n}, Q_{-n+1})$; otherwise remain at state Q_{-n} .
4. Continue until T defined as the first n such that at time $-n + 1$ each of these two paths accepts Q_{-n+1} . (Continue the Metropolis-Hastings algorithm forward to time zero using the α 's and Q 's from steps 1-3 to get the draw from π at time zero.)

By monotonicity, the upper path will accept a candidate point whenever the lower path will, and the two paths will be the same from that time forward. Consequently, our description of the upper process is a formality only, and, indeed, the upper process need not be run at all. Figure 1 illustrates a realization of the perfect IMH algorithm.

For sampling from complicated densities, we may wish to take advantage of the observation that neither the lowest state, l , nor the maximum value $\pi(l)/q(l)$ need be attained explicitly. If we are able to find a C such that

$$C \geq \frac{\pi(x)}{q(x)}, \quad \text{for all } x \tag{4}$$

then we know that

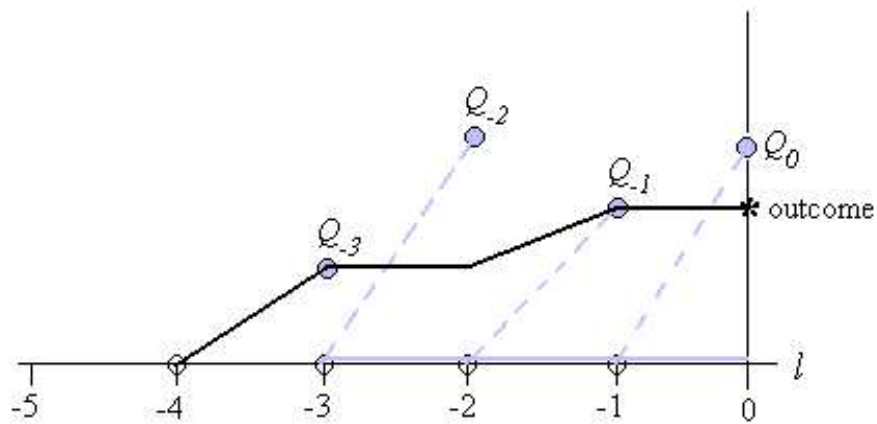
$$\alpha(l, y) \geq \frac{\pi(y)}{C q(y)},$$

so we could modify step 3(a) of the IMH algorithm to read

3. (a)' For the lower path: Accept a move from l to Q_{-n+1} at time $-n + 1$ with probability $\alpha(l, Q_{-n+1})$, otherwise remain at state l . That is, accept the move from l to Q_{-n+1} if $\alpha_{-n} \leq \frac{\pi(y)}{C q(y)}$.

We refer the reader to [3] for details on how one's choice of Q will affect the expected backward coupling time for the perfect IMH algorithm.

Figure 1: A Realization of the Perfect IMH Algorithm With Backward Coupling Time at -4



Dashed grey lines represent potential but unrealized arcs of the sample path. Solid grey lines represent the sample paths started at times -1, -2, and -3 that did not achieve the coupling. The solid black line represents the path whose outcome is ultimately observed in the perfect sampling algorithm.

3.3 An “Approximately Perfect” Algorithm

Identifying the “lowest point” l for the perfect IMH algorithm can sometimes be problematic. We now consider an imperfect variant of the perfect IMH algorithm where we use an approximate value for l . Even though this introduces error into the perfect IMH algorithm, preliminary empirical results suggest that the method is still superior to the traditional forward-time Metropolis-Hastings algorithm. Throughout, we refer to any perfect IMH algorithm with an approximated low point as an AIMH algorithm.

Recall that l is a point that maximizes the ratio

$$\frac{\pi(x)}{q(x)} \quad \text{or, equivalently} \quad \frac{h(x)}{q(x)}.$$

We could maximize this with either traditional deterministic methods or stochastic optimization methods. We choose the latter—a random search that may be built-in to the perfect IMH algorithm in a computationally expensive scenario. As some may view the built-in approach as too complicated to bother with, we also present the obvious approximate perfect IMH algorithm where l is approximated independently of the backward coupling algorithm. We wish to stress that in its simplest form, the AIMH algorithm is simply the IMH algorithm described in Section 3.2 where the maximum value of or an upper bound for the ratio π/q is approximated in any way convenient to the simulator, and that the following methods are only suggestions.

3.3.1 Independent Maximization

For some fixed N , we draw a sequence of values q_1, q_2, \dots, q_N from the distribution Q and approximate the low point by

$$\hat{l}_N = \arg \max_{x \in \{q_1, q_2, \dots, q_N\}} \frac{h(x)}{q(x)}.$$

We then proceed with the IMH algorithm as described in Section 3.2 using \hat{l}_N in place of l . Ideally, one would watch for a better estimate of l during the evaluation of h/q ratios in the IMH algorithm. Indeed, it is this advice that forms the basis for the built-in maximization described in Section 3.3.1.

We have assumed no knowledge of the structure of π or of h/q . Obviously if the region containing the maximizing point can be narrowed down, one should take advantage of this. One should also take advantage of more sophisticated deterministic and/or stochastic optimizers. In the following section, we present an approach in the spirit of parsimony, using only random deviates that must already be computable in order to run the perfect IMH algorithm.

3.3.2 Built-In Maximization

At each time step, we draw a candidate and compute the corresponding $\frac{\pi}{q}$ -ratio for the MH-algorithm. In a computationally expensive scenario, one may wish to use this information

for the stochastic search for the maximum of $\frac{\pi}{q}$. In the course of a simulation, if we find that the current candidate improves the current maximum, we immediately update our guess at this “low point”. To ensure progress in this search process, we go back at least N time steps (prescribed by the simulator a priori and adjusted based on convergence monitoring described in Section 3.4) for each MH-iteration. After these N time steps, we

1. stop if coupling (acceptance) has been achieved with respect to the current low point, recording the value output at time zero,
2. or, move back in time through another block of N time steps if coupling has not been achieved.

The algorithm

While the basic idea of the algorithm is very simple, unfortunately, its implementation is less straightforward.

We present pseudo-code showing that the different blocks of the algorithm can be collapsed into only two loops. We also wish to alert the reader that while we hope that the pseudo-code might be useful for someone wishing to implement the algorithm, we believe that it is too complicated to help in understanding the underlying idea.

We can basically split the algorithm into two parts – a backward (in time) and a forward (in time) loop.

1. Backward loop

- Initialize $\max = 0, t = 0, \text{coupled}=\text{FALSE}$.
- While $(t > -N)$ or (not coupled)
 - $t=t-1$
 - Draw a candidate $y_t \sim q(x)$ and $u_t \sim \text{unif}[0, 1]$.
 - Set $\text{cur} = \frac{\pi(y_t)}{q(y_t)}$.
 - If $\text{cur} > \max$
 - * $\max = \text{cur}$
 - * $\text{bct} = t$
 - * $\text{coupled}=\text{TRUE}$
 - * $x = y_t$
 - If $(t > -N)$ and $(u_t < \text{cur}/\max)$

- * bct = t
- * coupled=TRUE
- * $x = y_t$

2. Forward loop

- for $t = \text{bct} - 1$ to 1
 - $\text{acc} = \frac{\pi(x)q(y_t)}{q(x)\pi(y_t)}$
 - if $u_t < \text{acc}$
 - * set $x = y_t$

Remark Although the algorithm is used when we don't know the maximum, we still need to know that it exists – otherwise we are not guaranteed convergence.

Simulation Results

To assess the performance of the approximate IMH algorithm with a built-in maximization, we compare the output of this procedure to a regular forward IMH algorithm with approximately the same computational cost. To do so, we show the histograms of the draws from both algorithms with the target density superimposed.

In Figure 2, we wish to draw from a $N(4, 1)$ distribution using a double exponential candidate with rate 1 (i.e. $\pi(x) \propto e^{-x^2/2}$ and $q(x) \propto e^{-|x|}$ on $(0, \infty)$). After 100,000 draws, the average backward coupling time was 70 (the parameter N was set to 5), so the forward chain was run for 70 time steps for each draw to have comparable computational cost.

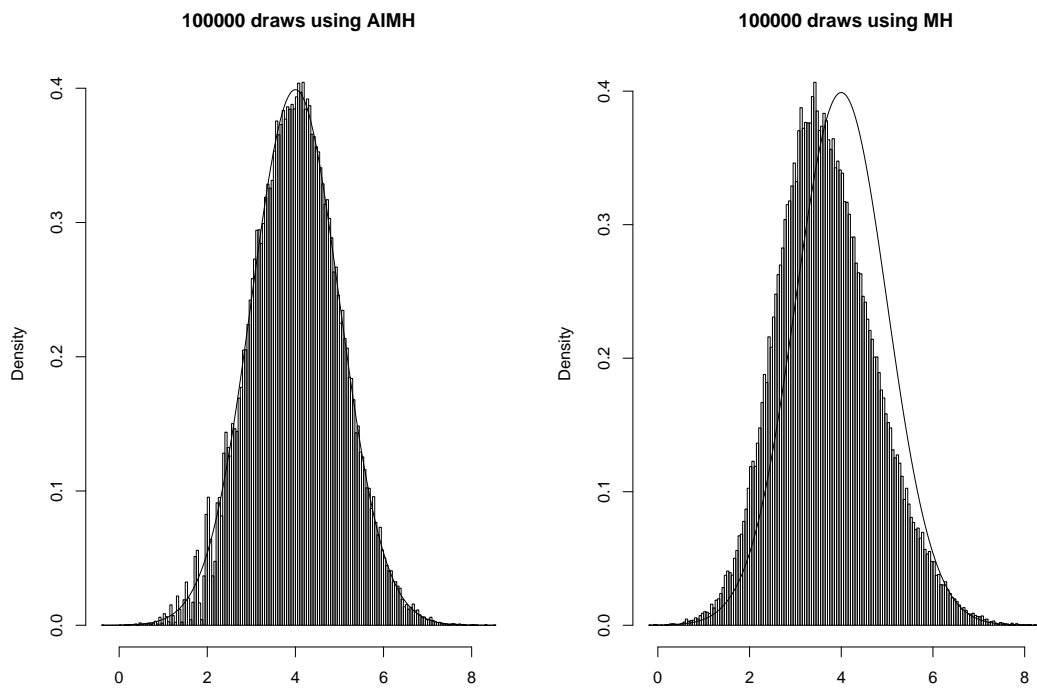
We also considered an example where the target and candidate density have bounded support, namely $\pi(x) \propto \exp\{-x\}|\sin(x)\cos(x)|$ and $q(x)$ the uniform density, both over the interval from 0 to 6. With these choices (and the parameter N set to 2), after 100,000 draws the average backward coupling was 5, so we ran the regular IMH algorithm forward for 5 time steps for each sample. The resulting histograms for the approximate IMH and the forward IMH algorithm are shown in Figure 3.

It appears that in both examples the AIMH algorithm clearly outperforms the the regular Metropolis-Hastings scheme.

3.4 A Backward Coupling Time Evaluation of the Approximate Maximization

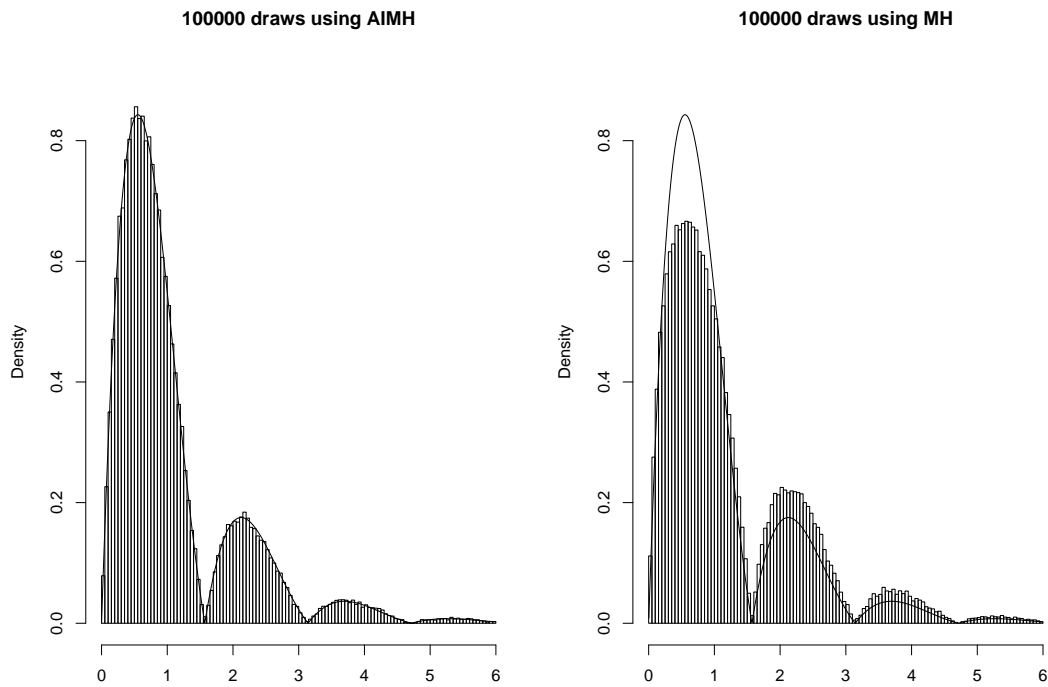
It is easy to see the effect of an using an incorrect (non-bounding) C in (4). Indeed, the perfect IMH algorithm is equivalent to rejection sampling, so if we do have a constant C such that

$$\max_x \frac{\pi(x)}{q(x)} \leq C,$$



The target density is $N(4, 1)$ with a double exponential candidate with rate 1. Histograms show 100,000 draws with an approximate IMH algorithm with built-in maximization and a regular Metropolis-Hastings procedure. The target density is superimposed.

Figure 2: Comparison of an approximate IMH algorithm with built-in maximization to an independent standard forward Metropolis-Hastings scheme.



The target density is $f(x) \propto \exp\{-x\}|\sin(x)\cos(x)|$ on the interval $(0,6)$ with a uniform candidate on $(0,6)$. Histograms show 100,000 draws with an AIMH algorithm with built-in maximization and a regular Metropolis-Hastings procedure. The target density is superimposed.

Figure 3: Comparison of an AIMH algorithm with built-in maximization to an independent standard forward Metropolis-Hastings scheme.

then the backward coupling time will be a geometric random variable with parameter

$$\begin{aligned}
p &:= P(\text{accept a move away from the low point } l) \\
&= \int P(\text{accept}|Y = y) q(y) dy \quad (Y \text{ is a candidate value}) \\
&= \int P\left(U \leq \frac{\pi(y)}{Cq(y)}\right) q(y) dy
\end{aligned}$$

where U is a uniform random variable on the interval from 0 to 1.

Now

$$\frac{\pi(y)}{q(y)} \leq C \quad \text{for all } y \quad \Rightarrow \quad \frac{\pi(y)}{Cq(y)} \leq 1,$$

so we have that

$$\begin{aligned}
p &= \int \frac{\pi(y)}{Cq(y)} \cdot q(y) dy \\
&= \frac{1}{C} \int \pi(y) dy = \frac{1}{C}.
\end{aligned}$$

Therefore, the expected backward coupling time for this geometric random variable is $1/p = C$.

Now suppose that we approximate C by C' . That is, we believe that

$$\frac{\pi(x)}{q(x)} \leq C' \quad \text{for all } x,$$

when it is actually the case that

$$\frac{\pi(x)}{q(x)} \leq C' \quad \text{for some } x, \quad \text{but also} \quad \frac{\pi(x)}{q(x)} > C' \quad \text{for some } x.$$

Presumably, we have done well in our approximation and would like to believe that the x for which $\pi(x)/q(x) > C'$ comprise only a small set, although this is not a necessary assumption.

If we have truly found a C' that bounds π/q , we would expect, in simulation, to see a backward coupling time of C' time steps. In reality, we may not have that $\pi(y)/C'q(y) \leq 1$ and so we will see a backward coupling time which is geometric with parameter

$$\begin{aligned}
p' &= P(\text{accept a move away from the “false” low point } l') \\
&= \int P(\text{accept}|Y = y) q(y) dy \\
&= \int P\left(U \leq \min\left(1, \frac{\pi(y)}{C'q(y)}\right)\right) q(y) dy \\
&= \frac{1}{C'} \int_{A_1} \pi(y) dy + \int_{A_1^c} q(y) dy,
\end{aligned}$$

where

$$A_1 = \left\{ y : \frac{\pi(y)}{q(y)} \leq C' \right\}$$

and A_1^c is the complement of A_1 .

The average backward coupling time that we would observe is then

$$\frac{1}{p'} = C' [\Pi(A_1) + C' \cdot Q(A_1)]^{-1} < C'$$

where Π and Q are the distribution functions associated with the densities π and q , respectively.

Intuitively, we will observe an average backward coupling time that is smaller than we expect since we are accepting moves away from the “low point” more often than we should. That is, we didn’t use the highest value of π/q which translates to that we are not using the lowest possible acceptance probability.

Example:

As a simple illustrative example, we consider the exponential target and candidate densities $\pi(x) = 3e^{-3x}$ and $q(x) = 2e^{-2x}$ for $x > 0$. Here

$$\sup_{x>0} \frac{\pi(x)}{q(x)} = \frac{3}{2},$$

so we take $C = 3/2$. In Table 1, we show the average backward coupling time obtained in 100,000 draws from π using various correct and incorrect values for C' increasing from $C' = 0.1$ to $C' = 3.0$ by increments of 0.1. For example, when we take C' to be 0.1, we observe an average backward coupling time of 1.00139. If we really had a bounding C' , we should see an average backward coupling time of 0.1. Hence, we know that 0.1 is not an upper bound for π/q . The results Table 1 suggests, as expected, that we may take C' to be 1.5 or greater. (At first glance it appears that 1.4 might be an acceptable upper bound for π/q , but a confidence interval for the true mean backward coupling time based on this large sample of size 100,000 using the sample variance of backward coupling times did not include 1.40.)

4 An Adaptive IMH Algorithm

It is likely that any person who has ever given a presentation on the Metropolis-Hastings algorithm, perfect or otherwise, has been asked the question:

How do you choose the candidate distribution?

We usually find ourselves giving the unsatisfactory two-part response:

Choose a candidate distribution Q with density q so that

1. q is overdispersed with respect to the target density π , and
2. you are able to simulate (draw) values from Q .

Table 1: Finding the True C by Examining backward Coupling Times for 100,000 Draws from $\pi(x) = 3e^{-3x}$ Using $q(x) = 2e^{-2x}$, True $C = 1.5$

C'	Mean BCT	C'	Mean BCT
0.1	1.00139	1.6	1.59841
0.2	1.00583	1.7	1.70207
0.3	1.01353	1.8	1.79753
0.4	1.02398	1.9	1.89421
0.5	1.03962	2.0	1.99779
0.6	1.05577	2.1	2.10254
0.7	1.07876	2.2	2.19480
0.8	1.10573	2.3	2.29574
0.9	1.13626	2.4	2.39975
1.0	1.17377	2.5	2.49925
1.1	1.22025	2.6	2.59338
1.2	1.27262	2.7	2.71045
1.3	1.33592	2.8	2.81186
1.4	1.40805	2.9	2.90070
1.5	1.49815	3.0	2.99516

We realize that this is not the most helpful answer. There do, in fact, exist “self-targeting” approaches such as in [15, 16], and we now propose, in a similar spirit, an “adaptive” algorithm that will create a candidate for the user. We begin with a motivating example.

Suppose we wish to draw values from the distribution with density

$$\pi(x) \propto e^{-x} |\sin(x) \cos(x)|, \quad \text{for } x > 0.$$

Let us further suppose that we choose a $\Gamma(5, 1/2)$ candidate. That is, we will draw candidate values from the distribution with density

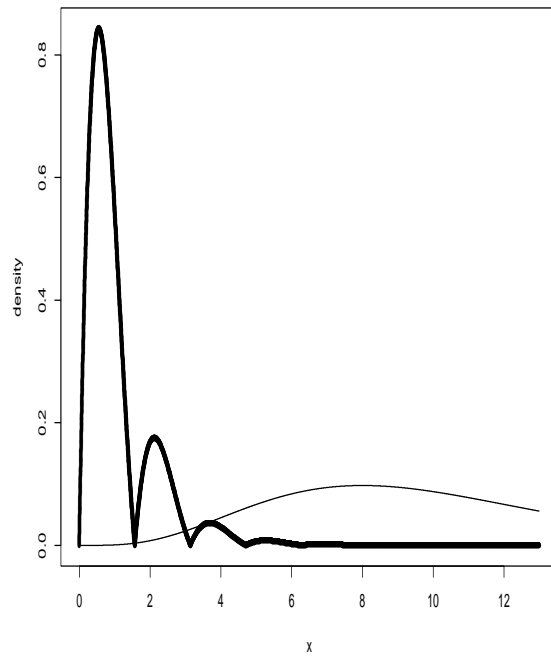
$$q(x) = \frac{1}{\Gamma(5)} \left(\frac{1}{2}\right)^5 x^4 e^{-\frac{1}{2}x}, \quad \text{for } x > 0.$$

In many cases, one can still get decent results from the Metropolis-Hastings algorithm even with a “bad” candidate density. We have chosen the (strange) target density since it is a multi-modal example and have chosen a candidate density which not only fails condition (2) but is “very bad” in the sense that the bulk of the mass is far from that for the target density. Both densities are shown together in Figure 4.

In Figure 5 we give histograms showing the resulting distribution of values given by 100,000 draws using a standard Metropolis-Hastings algorithm run forward for 300, 500, 1000, and 2000 time steps.

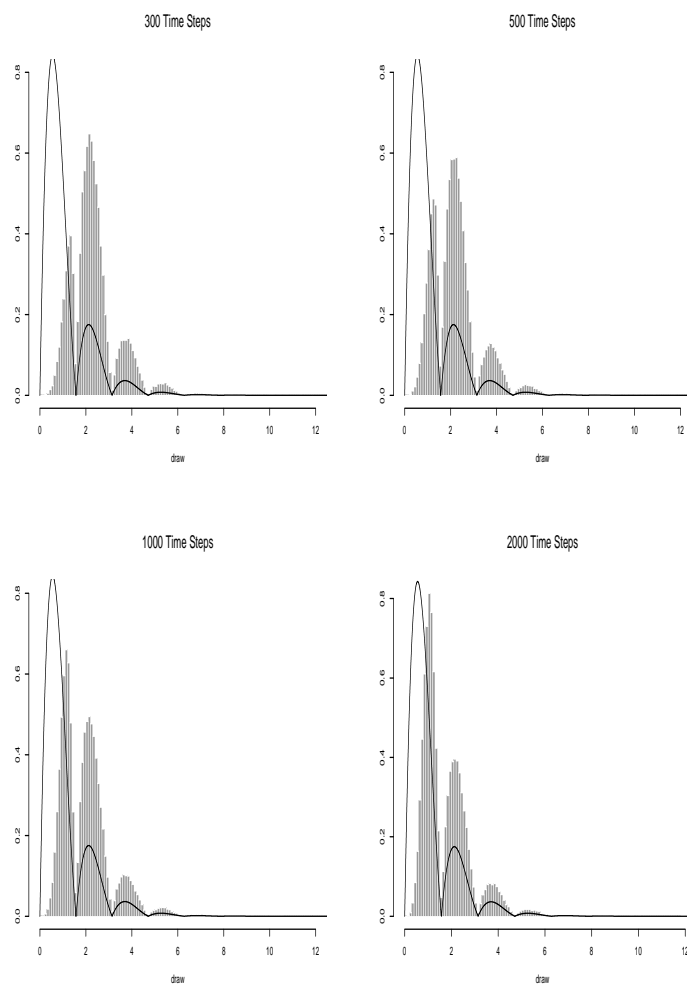
As expected, we are not achieving convergence to the target distribution. Indeed, there is not much difference at all between 1000 and 2000 time steps. However, we would like to not

Figure 4: The Target and Candidate Densities



The multi-modal density is the normalized target density π . The unimodal density is the candidate density q .

Figure 5: 100,000 Draws from $\pi(x) \propto e^{-x} |\sin(x) \cos(x)|$ Using the Metropolis-Hastings Algorithm with the $\Gamma(5, 1/2)$ Candidate



waste the information gained after 300 (or fewer) time steps— as we have at this point at least made some progress in moving towards the target distribution and we have certainly begun to capture its multi-modal behavior. We propose to start over with the Metropolis-Hastings algorithm using the output of our previous failed attempt as a candidate distribution. This involves representing the output of our previous attempt in the form of a histogram to a desired accuracy (bin width), and drawing values from this distribution of values. Note that we are now taking the candidate density q to be a step function that is constant over each bin, but that q is still a continuous density.

Consider the output after 300 time steps depicted in the upper left plot of Figure 5. Any distribution estimated with a finite sample will have a finite support. In this particular case, if we were to make a “refinement” to the Metropolis-Hastings algorithm by drawing candidate values from this histogram distribution, we would never get proposals that are less than 0.4 or greater than approximately 6.0. For this reason, we must “spread out” the candidate histogram distribution. We illustrate the procedure with a simplified histogram.

Spreading the Candidate Distribution

Suppose that we are trying to ultimately draw values from a continuous target distribution with density π with support $(0, \infty)$ and that we begin with an independent candidate distribution with density q . Let us further suppose that we have already run the Metropolis-Hastings for some $T > 0$ time steps and that this has resulted in a sample with the histogram depicted in Figure 6.

We have chosen, in this case, to store the output in a histogram with bin width 0.1. We store this width and the vector of histogram heights:

$$\begin{aligned} \text{binwidth} &= 0.1 \\ \text{histogram} &= [0.0, 0.0, 2.5, 3.5, 0.0, 3.0, 1.0]. \end{aligned}$$

We “spread” this histogram over the entire support set as follows.

1. For gaps on the left end, we put a mass that is equal to that for the first non-empty bin encountered when moving from left to right. In this example, the histogram vector becomes

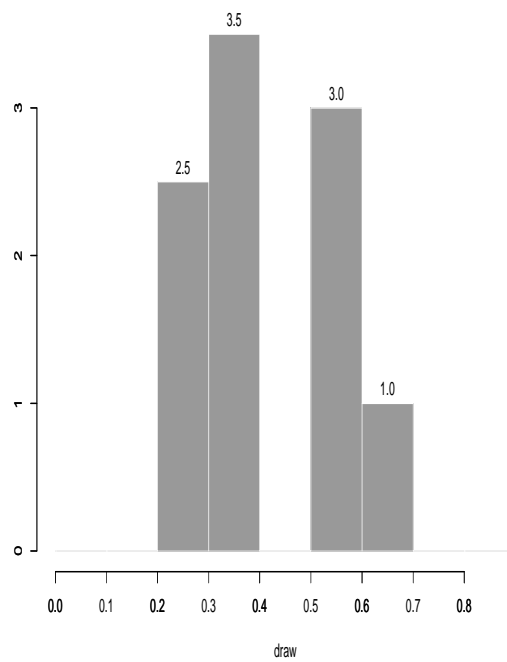
$$\text{histogram} = [2.5, 2.5, 2.5, 3.5, 0.0, 3.0, 1.0].$$

2. For gaps in the middle, we put a mass that is equal to the average mass of the two surrounding bins. In this example, the histogram vector becomes

$$\text{histogram} = [2.5, 2.5, 2.5, 3.5, 3.25, 3.0, 1.0].$$

(Note: If there had been more empty bins between the 3.5 and the 3.0, then they would have all been assigned the height of 3.25.)

Figure 6: Output of the MH Algorithm run for $T > 0$ Time Steps With Candidate q



3. We create a “dummy bin” on the right end that, in this case, will represent the probability of drawing a value greater than 0.7. We do this by putting a mass equal to that for the last non-empty bin. Our histogram vector becomes

$$\text{histogram} = [2.5, 2.5, 2.5, 3.5, 3.25, 3.0, 1.0, 1.0].$$

4. Finally, we renormalize the histogram. In this case, with a bin width of 0.1, the heights should add up to 10. Since they add up to 19.25, we multiply all values by $10/19.25$ to get

$$\text{histogram} = [1.2987013, 1.2987013, 1.2987013, 1.8181818, \\ 1.6883117, 1.5584416, 0.5194805, 0.5194805].$$

Drawing from the Candidate Histogram Distribution

1. We begin by turning the histogram heights into probabilities. We create the vector

$$\begin{aligned} \text{probabilities} &= \text{binwidth} * \text{histogram} \\ &= [0.12987013, 0.12987013, 0.12987013, 0.18181818, \\ &\quad 0.16883117, 0.15584416, 0.05194805, 0.05194805]. \end{aligned}$$

2. We select a bin by drawing a random value U_1 from the Uniform[0,1] distribution. In this case if $U_1 < 0.12987013$, we select the first bin. If $U_1 < 0.12987013 + 0.12987013$, we select the second bin, and so on.
3. (a) If we have selected any bin but the last, we then draw our candidate value uniformly from within this bin. For example, if we have selected the fourth bin, we draw a value U_2 from the Uniform[0,1] distribution and set

$$\text{candidate} = \text{binwidth} * U_2 + (4 - 1) * \text{binwidth}.$$

- (b) If we have selected the last bin, we draw a value from a distribution with support (c, ∞) where c is the upper bound for the original histogram. In this example, $c = 0.7$.

We have chosen to use a simple shifted exponential distribution with a rate 1. (This rate was chosen so that this exponential piece of the candidate extends in a continuous way off of the last bar of the histogram.) In this case, we draw a value from an exponential distribution with rate 1 that is shifted 0.7 units to the right.

To summarize, in this example our candidate density is

$$q(x) = \begin{cases} 1.2987013 & , \text{ for } 0 < x < 0.1 \\ 1.2987013 & , \text{ for } 0.1 \leq x < 0.2 \\ 1.2987013 & , \text{ for } 0.2 \leq x < 0.3 \\ 1.8181818 & , \text{ for } 0.3 \leq x < 0.4 \\ 1.6883117 & , \text{ for } 0.4 \leq x < 0.5 \\ 1.5584416 & , \text{ for } 0.5 \leq x < 0.6 \\ 0.5194805 & , \text{ for } 0.6 \leq x < 0.7 \\ 0.5194805 e^{-(x-0.7)} & , \text{ for } x \geq 0.7 \end{cases}$$

We now return to our example.

Example:

We will use the adaptive IMH algorithm to draw values from

$$\pi(x) \propto e^{-x} \cdot |\sin(x) \cos(x)|, \quad x > 0$$

using the $\Gamma(5, 1/2)$ candidate distribution with density

$$q(x) = \frac{1}{\Gamma(5)} \left(\frac{1}{2}\right)^5 x^4 e^{-\frac{1}{2}x}, \quad x > 0$$

as our starting candidate.

Running a standard Metropolis-Hastings algorithm forward for 100 time steps produces the distribution of values shown in Figure 7. The target density is also shown.

Using the histogram shown in Figure 7 as the new candidate density, we again run the standard Metropolis-Hastings algorithm forward for 100 time steps. The output is after this “refinement” shown in Figure 8.

Finally, using the histogram shown in Figure 8 as the new candidate density, we run the standard Metropolis-Hastings algorithm forward for 100 time steps. The output is after this second refinement is shown in Figure 9. At this point, it appears, at least visually, that we have achieved convergence. Standard convergence assesment techniques measuring forms of “ distance” between the target density and the simulation output may be applied.

5 Conclusions

We have given an imperfect variant of a perfect simulation algorithm for which preliminary research shows substantial empirical evidence of usefulness in terms of accuracy and speed over traditional non-perfect Markov chain Monte Carlo algorithms. We have also given a self-targeting approach to generate a candidate density for a standard Metropolis-Hastings

Figure 7: 100,000 Draws from π After 100 Time Steps Using $Q \sim \Gamma(5, 1/2)$

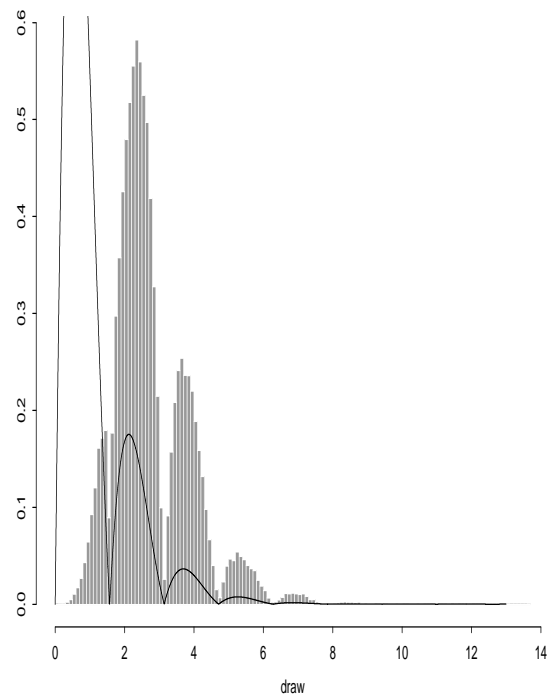


Figure 8: Refinement 1: 100,000 Draws from π After 100 Time Steps Using the Distribution Depicted in Figure 7 as a Candidate

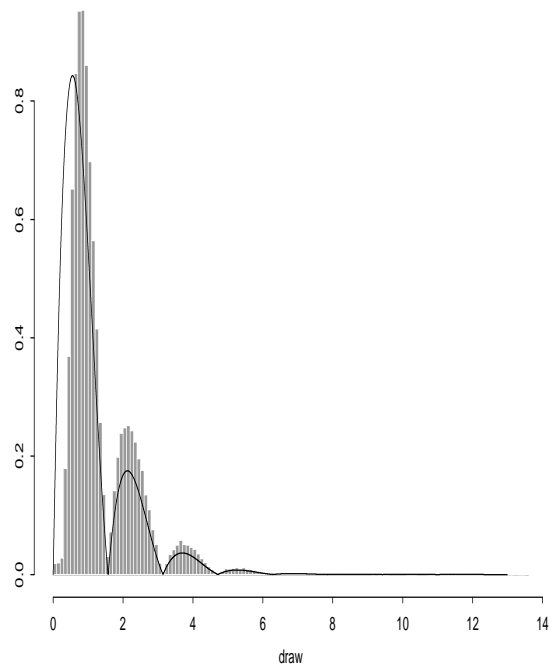
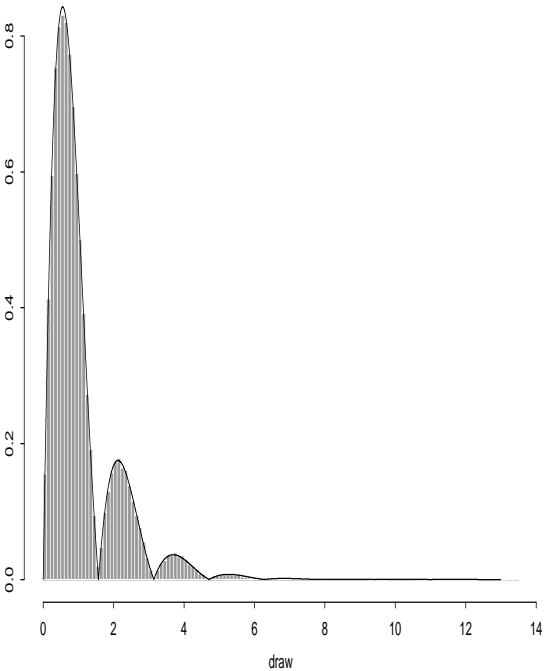


Figure 9: Refinement 2: 100,000 Draws from π After 100 Time Steps Using the Distribution Depicted in Figure 8 as a Candidate



algorithm that, due to the forced “spreading” of the evolving candidate distribution, will not “get stuck” at a fixed point or in a fixed incorrect distribution. Analytic support for these promising algorithms are the subject of ongoing research.

References

- [1] H. Cai. Exact bound for the convergence of metropolis chains. 1997.
- [2] G. Casella, M. Lavine, and C. Robert. Explaining the perfect sampler. Working Paper 00-16, State University of New York at Stony Brook, Duke University, Durham., 2000.
- [3] J.N. Corcoran and R.L. Tweedie. Perfect sampling from Independent Metropolis-Hastings Chains. *Journal of Statistical Planning and Inference*, 104:297–314, 2002.
- [4] J.A. Fill. An interruptible algorithm for perfect sampling via Markov chains. *ANNAP*, 8:131–162, 1998.
- [5] S.G. Foss and R.L. Tweedie. Perfect simulation and backward coupling. *Stochastic Models*, 14:187–203, 1998.
- [6] S.G. Foss, R.L. Tweedie, and J.N. Corcoran. Simulating the invariant measures of Markov chains using horizontal backward coupling at regeneration times. *Prob. Eng. Inf. Sci.*, 12:303–320, 1998.
- [7] O. Häggström, M.N.M. van Liesholt, and J. Møller. Characterisation results and Markov chain Monte Carlo algorithms including exact simulation for some spatial point processes. *Bernoulli*, 5:641–659, 1999.
- [8] W. K. Hastings. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.
- [9] W.S. Kendall. Perfect simulation for the area-interaction point process. In L. Accardi and C.C. Heyde, editors, *Probability Towards the Year 2000*, pages 218–234. Springer, New York, 1998.
- [10] K.L. Mengersen and R.L. Tweedie. Rates of convergence of the Hastings and Metropolis algorithms. *Annals of Statistics*, 24:101–121, 1996.
- [11] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.
- [12] J. Møller. Perfect simulation of conditionally specified models. *Journal of the Royal Statistical Society, Ser. B*, 61(1):251–264, 1999.
- [13] D.J. Murdoch and P.J. Green. Exact sampling from a continuous state space. *Scandinavian Journal of Statistics*, 25:483–502, 1998.

- [14] J.G. Propp and D.B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9:223–252, 1996.
- [15] O. Stramer and R.L. Tweedie. Langevin-type models ii: Self-targeting candidates for mcmc algorithms. *Methodology and Computing in Applied Probability*, 1:307–328, 1999.
- [16] Osnat Stramer and Richard L. Tweedie. Self-targeting candidates for hastings-metropolis algorithms. Technical Report 261, Department of Statistics, The University of Iowa, 1997.
- [17] L. Tierney. Markov chains for exploring posterior distributions (with discussion). *Annals of Statistics*, 1:1701–1762, 1994.