LMS J. Comput. Math. 13 (2010) 246–259 © 2010 Authors
doi:10.1112/S1461157007000022

Perfect posterior simulation for mixture and hidden Markov models

Kasper K. Berthelsen, Laird A. Breyer and Gareth O. Roberts

Abstract

In this paper we present an application of the read-once coupling from the past algorithm to problems in Bayesian inference for latent statistical models. We describe a method for perfect simulation from the posterior distribution of the unknown mixture weights in a mixture model. Our method is extended to a more general mixture problem, where unknown parameters exist for the mixture components, and to a hidden Markov model.

1. *Introduction*

Following the seminal work in [7] on perfect simulation, considerable effort has gone into developing the general methodology and extending the range of distributions to which these methods apply. In this paper, we describe the implementation of a perfect simulation algorithm for generating samples from the posterior distribution in two different Bayesian latent-variable models. Our methods are applied first to the posterior distribution of the unknown mixture weights in a Bayesian analysis of a mixture of known distributions. Secondly, we consider a Bayesian analysis of a hidden Markov model.

Our approach will be to use and extend a collection of techniques for perfect simulation. Central to all of our methodology will be the use of the read-once coupling from the past (roCFTP) algorithm for perfect simulation; see [9]. At the core of this algorithm is the construction of uniformly ergodic blocks of Markov chain update rules. The main challenge is to construct these blocks of updates so that they have a significant coalescence probability, that is, the probability that a block maps the entire state space into just one state should be non-negligible.

The main example in this paper is the mixture problem where the mixture components are known but the mixture weights are unknown. For this example, the primary updating construction makes use of the duality principle (see [5]) by augmenting allocation variables. This allows the simultaneous update of all potential allocations within a rectangular region in the allocation space and, crucially, enables us to determine a rectangular region containing the updated allocations. In some situations, these bounds turn out to be too 'sloppy' for our purposes. This leads us to consider 'exact' bounds that circumvent the problem of sloppy bounds. On the other hand, these exact bounds are comparatively more expensive in terms of computational complexity.

As an extension of our methodology, we consider a hidden Markov chain setup. In this setup, allocation variables are no longer a priori identically distributed but, rather, are distributed according to a Markov chain. Posterior simulation is done using the same basic approach as for the mixture problem.

Received 9 January 2007; revised 3 April 2009.

2000 Mathematics Subject Classification 60J22 (primary), 62F12 (secondary).

This research was supported by the European Union TMR network ERB-FMRX-CT96-0095 on 'Spatial and Computational Statistics', by an Engineering and Physical Sciences Research Council grant, and by the Danish Natural Science Research Council (grant no. 272-06-0442).

There exist several works on perfect simulation in connection with mixture models. In [6], the basic update function was used in a CFTP-type algorithm for perfect simulation; the detection of coalescence was done by tracking a bounding set, similar to what we do. Unlike our method, however, the method of [6] seems limited to mixtures of at most three components. In [4] a perfect slice sampler was presented, which turns out to be inefficient for moderately sized data sets.

The remainder of the paper is organised as follows. In Section 2 we specify the basic mixture problem and review the roCFTP algorithm. In Section 3 we construct an update function that has the posterior of interest as its equilibrium distribution. Section 4 is concerned with how to construct bounds for the image of this update function. These bounds are then used in Section 5 to produce perfect samples of the posterior weight distribution. Furthermore, Section 5 contains a simulation study comparing different ways of combining the bounds developed in Section 4. In Section 6 we describe an algorithm for perfect simulation from the posterior transition probabilities in a two-state hidden Markov model.

2. *Problem specification*

2.1. *The posterior weight distribution*

Assume that the data points $\eta_1, \ldots, \eta_n$ are an independent sample from an r-component mixture

$$\sum_{k=1}^{r} m_k p_k(\cdot),$$

where the component densities $p_1, \ldots, p_r$ are assumed to be known and the mixture weights $m_1, \ldots, m_r$ are subject to the restrictions $m_k \geqslant 0$ and $\sum_{k=1}^{r} m_k = 1$.

Upon introducing a uniform Bayesian prior distribution (Dirichlet $\mathcal{D}(1, 1, \ldots, 1)$) for the unknown weights $m = (m_1, \ldots, m_r)$ on the simplex

$$S = \left\{ (m_1, \ldots, m_r) : m_k \geqslant 0 \text{ for all } k \text{ and } \sum_{k=1}^{r} m_k = 1 \right\},$$

we obtain the posterior distribution

$$\pi(m_1, \ldots, m_r \mid \eta) = \prod_{i=1}^{n} \left[\sum_{k=1}^{r} m_k p_k(\eta_i) \right]. \tag{2.1}$$

As this is typically intractable, it is usual to resort to Monte Carlo methods to explore this distribution. Mixing of Markov chain Monte Carlo (MCMC) algorithms for this problem is often problematical. Our goal in perfect simulation is to avoid the burn-in problem associated with ordinary MCMC approaches.

2.2. *Wilson's read-once CFTP algorithm*

The perfect simulation algorithms considered in this paper are all examples of the roCFTP algorithm introduced in [9]. For completeness, we briefly review the roCFTP algorithm below and define some of the related notation that will be used throughout this paper.

Assume that we want to sample from a distribution Π on a state space Ω and that we know how to generate a sequence of independent realisations of a random update function $C : \Omega \to \Omega$ with the properties that: (1) it preserves stationarity, that is, $\int_\Omega \mathbb{P}[C(x) \in A]\Pi(dx) = \Pi(A)$ for all $A \subseteq \Omega$; and (2) it has a positive probability of being coalescent, that is, of mapping the entire state space into a single state. A realisation of a random update function is denoted by an update function. In what follows, $\#W$ will denote the cardinality of the set W.

Under the above assumptions, we can generate a perfect sample as follows. Generate a sequence $C_1, C_2, \ldots$ of independent realisations of the random update function C. Let T_i denote the index of the ith coalescent update function; that is, C_{T_i} is coalescent and hence $C_{T_i}(x)$ does not depend on x. Then $x_1 = C_{T_2 - 1} \circ \ldots \circ C_{T_1}(x)$ does not depend on $x \in \Omega$ and, more importantly, x_1 is a sample from Π. In general, if $x_i = C_{T_{i+1}-1} \circ \ldots \circ C_{T_i}(x)$, then $x_1, x_2, \ldots$ are independent and identically distributed according to Π. For a proof based on the ideas of Propp and Wilson's coupling from the past algorithm, see [9]; for a more algebraic proof, see [3].

In most cases of interest, the random update function C is a compound made up of several 'basic' random update functions that each satisfy (1) but not necessarily (2). So $C_i = F_{i,K} \circ \ldots \circ F_{i,2} \circ F_{i,1}$ where $F_{i,1}, \ldots, F_{i,K}$ are random update functions that preserve stationarity. For example, F could represent a Gibbs update of the type described in Section 3.

A further complication is that typically there is no feasible way of telling whether a given realisation C_i is coalescent or not. In some situations it may be possible to find a criterion which implies that C_i is coalescent but not the other way round. If such a criterion is fulfilled, we can declare C_i coalescent. Assume that we are given such a criterion for coalescence. We then redefine T_i to be the index of the ith update function for which the criterion is satisfied. Letting x_i be defined as before, we obtain a subsample of the original perfect sample which, crucially, is still a perfect sample from Π; see [3]. So it would seem that the better the criterion is at detecting coalescent update functions, the more efficient our perfect sampler will be. The downside is that an effective criterion can come at a high computational cost.

Regarding the detection of coalescence, if Ω is of sufficiently low cardinality, it may be feasible to check whether each $x \in \Omega$ results in the same value of $C_i(x)$. In this case, all coalescent update functions will be detected. When this is not feasible, we shall attempt to detect coalescence by using bounding sets as follows. Recall that $C_i = F_{i,K} \circ \ldots \circ F_{i,2} \circ F_{i,1}$. For each $F_{i,t}$ with $t = 1, \ldots, K$, assume that given $W \subseteq \Omega$ we can construct a bounding set W' so that $F_{i,t}(W) \subseteq W'$, where $F_{i,t}(W) = \{F_t(x) : x \in W\}$. For each C_i we initialise by setting $W = \Omega$. The ability to construct bounding sets then allows us to update W sequentially as we apply each $F_{i,t}$. If after applying the last update function $F_{i,K}$ we have $\#W' = 1$, we declare C_i to be coalescent; otherwise, we declare C_i non-coalescent. In the remainder of this paper, whenever $W' = F_t(W)$ we say that W' is an exact bounding set. The roCFTP perfect simulation algorithm using bounding sets is illustrated in Figure 1 and can be summarised in pseudo-code as follows.

```
1. Choose arbitrary $x$ in $\Omega$ and set $s := 0$
2. For $i = 1, 2, 3, \ldots$
3.     Set $W := \Omega$ and $x_{\text{old}} := x$
4.     For $t = 1, \ldots, K$
5.         Generate $F_t$ and set $x := F_t(x)$
6.         Determine $W' \subseteq \Omega$ such that $F_t(W) \subseteq W'$
7.         Set $W := W'$
8.     If $\#W = 1$, set $x_s := x_{\text{old}}$ and $s := s + 1$
```

Notice that the x_0 generated by the algorithm is not part of the sample from Π; only $x_1, x_2, \ldots$ are. Notice further that each update function F_t is used only once, in contrast to conventional CFTP algorithms, hence the name 'read-once' CFTP.

In practice, it is generally easy to construct F_t so that it preserves stationarity. In some cases we require more. If Ω is an unbounded state space, we typically want F_1 to map Ω into a bounded region with positive probability. This turns out not to be a problem here. Still, we need F to be constructed in a way that allows the construction of bounding sets.

In Section 3 we shall consider how to construct a random map F that has (2.1) as its equilibrium distribution. In Section 4 we show how to construct bounding sets, making perfect simulation feasible.

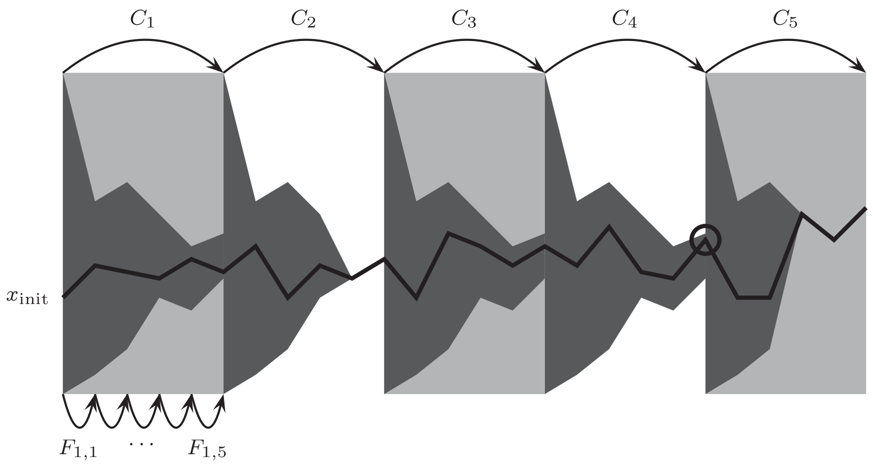

FIGURE 1. *Illustration of the roCFTP algorithm. The dark grey regions correspond to bounding sets. The solid line corresponds to the target process started in the arbitrarily chosen state x_{init}. In this example the compound update functions C_2 and C_5 are declared coalescent; accordingly, $T_1 = 2$, $T_2 = 5$ and $x_1 = C_4 \circ C_3 \circ C_2(x)$ (circled) is a sample from Π.*

3. *The update function: Gibbs sampler updates*

It is well known that the Gibbs sampler can be used to approximately simulate from the posterior distribution (2.1) by means of the data augmentation methodology; see [5]. More precisely, to simulate from π we define auxiliary variables $Z_1, \ldots, Z_n \in \{1, \ldots, r\}$, which represent the component allocations of each data point, and use a Gibbs sampler, whose updates are formally

$$\begin{aligned}(M_1', \ldots, M_r') &\sim \pi_0(\cdot \mid Z_1, \ldots, Z_n),\\ Z_i' &\sim \pi_i(\cdot \mid M_1', \ldots, M_r') \quad \text{for } i = 1, \ldots, n,\end{aligned}$$

where $\pi_0(\cdot \mid z_1, \ldots, z_n) = \mathcal{D}(N_1(z) + 1, \ldots, N_r(z) + 1)$ with $N_k(z) = \#\{s : z_s = k\}$ and

$$\pi_i(k \mid m_1, \ldots, m_r) = m_k p_k(\eta_i) \Big/ \sum_{j=1}^{r} m_j p_j(\eta_i) \quad \text{for } k = 1, \ldots, r. \tag{3.1}$$

Recall that the Dirichlet distribution $\mathcal{D}(\alpha_1 + 1, \ldots, \alpha_r + 1)$ has density on S given by

$$h_\alpha(m_1, \ldots, m_r) = \frac{\Gamma(r + \alpha_1 + \cdots + \alpha_r)}{\Gamma(1 + \alpha_1) \cdots \Gamma(1 + \alpha_r)} m_1^{\alpha_1} \ldots m_r^{\alpha_r}.$$

Implementation of the above Gibbs sampler involves a recursively defined sequence

$$X_t = (Z_{1t}, \ldots, Z_{nt}, M_{1t}, \ldots, M_{rt})$$

such that $X_{t+1} = F_{t+1}(X_t)$, where $F_1, F_2, \ldots$ are independent and identically distributed random functions of the form

$$F(x) = (Z_1'(x), \ldots, Z_n'(x), M_1'(x), \ldots, M_r'(x)). \tag{3.2}$$

We now give the details of the construction of F.

To generate a probability vector $(M_1'(x), \ldots, M_r'(x))$ with the Dirichlet $\mathcal{D}(N_1(x) + 1, \ldots, N_r(x) + 1)$ distribution, it suffices to generate independent random variables

$$G_1 \sim \Gamma(N_1(x) + 1), \ldots, G_r \sim \Gamma(N_r(x) + 1),$$

where $\Gamma(N+1)$ denotes a gamma distribution with shape parameter $N+1$ and scale parameter 1, and then set

$$M'_k(x) = G_k(x) \Big/ \sum_{j=1}^{r} G_j(x).$$

To generate each $Z'_s(x)$, we suggest using the following sequential rejection method. Generate r independent uniform random variables $\xi_{s,1}, \ldots, \xi_{s,r} \sim \mathrm{U}[0,1]$. Then, for each component $k = 1, \ldots, r$ in turn, we accept $Z'_s(x) = k$ if:

- $p_k(\eta_s) M'_k(x) / \sum_{j=k}^{r} p_j(\eta_s) M'_j(x) > \xi_{s,k}$; and
- no other component $j < k$ has yet been accepted.

This method of generating $Z'_s(x)$ has the advantage of requiring a bounded number of iterations to return an answer, independently of the weights $M'_j(x)$ or the factors $p_k(\eta_s)$.

3.1. *Generating gamma random variables*

In the remainder of the paper, we need to simulate $G_k \sim \Gamma(N_k(x)+1)$ simultaneously for a whole range of values of $N_k(x)$. As the distribution of G_k depends on x only through $N_k(x)$, we will sometimes use the notation $G_k(N_k)$.

In this section we specify how to generate a monotone random function $G : \{a, a+1, \ldots, b\} \to \mathbb{R}$, where $a, b \in \mathbb{N}$ and $1 \leqslant a \leqslant b \leqslant n$, such that $G(i) \sim \Gamma(i)$ and $G(i) \leqslant G(i+1)$ for $i = a, \ldots, b-1$. The construction of G consists of two steps. First, we construct the skeleton $\{J^{(s)}, H^{(s)} : s = 1, \ldots, L\}$ of G, where $\{J^{(i)}\} \subset \mathbb{N}$ and $\{H^{(i)}\} \subset \mathbb{R}$ are two random and strictly increasing sequences of random length L. Second, given the skeleton, the random function G is constructed by setting, for $i = a, \ldots, b$, $G(i) = H^{(s)}$ whenever $J^{(s)} \leqslant i < J^{(s+1)}$. In the following, let $g(\cdot\,; i)$ denote the density of a gamma distribution with shape parameter i and scale parameter 1.

The skeleton is constructed as follows.

```
1.  Generate $\gamma \sim \Gamma(a)$
2.  Set $J^{(1)} := a$ and $G^{(1)} := \gamma$
3.  Conditionally on $\gamma$ generate $u \sim \mathrm{U}[0, g(\gamma; a)]$
4.  Find the largest $i \geqslant a$ such that $u \leqslant g(\gamma; i)$
5.  Set $s := 1$
6.  Repeat steps 7 to 10 until $i \geqslant b$
7.      Set $s := s + 1$
8.      Generate $(\gamma, u)$ uniformly from the region $\{(\gamma, u) : g(\gamma; i) < u \leqslant g(\gamma; i+1)\}$
9.      Given $(\gamma, u)$ find the largest $i'$ such that $u \leqslant g(\gamma; i')$
10.     Set $J^{(s)} := i + 1$, $G^{(s)} := \gamma$ and $i := i'$
11. Set $L := s$ and $J^{(L+1)} := b + 1$
```

A proof of the correctness of this algorithm is given in Appendix A.

In practice, we sample (γ, u) uniformly from the region $\{(\gamma, u) : g(\gamma; i) < u \leqslant g(\gamma; i+1)\}$ by first sampling γ from a density proportional to $(g(\gamma - i; i+1) - g(\gamma - i; i))\mathbb{1}[\gamma \geqslant i]$ and then, conditional on γ, sampling u from $\mathrm{U}[g(\gamma; i), g(\gamma; i+1)]$. For simulating γ, we introduce a shifted version $\tilde{\gamma} \equiv \gamma - i$ and observe that

$$g(\tilde{\gamma}; i+1) - g(\tilde{\gamma}; i) \propto (1 + \tilde{\gamma}/i)^{i-1} e^{-\tilde{\gamma}} \tilde{\gamma}/i \quad \text{for } \tilde{\gamma} \geqslant 0, \tag{3.3}$$

where the right-hand side of (3.3) is a normalised probability density. Furthermore,

$$(1 + \tilde{\gamma}/i)^{i-1} e^{-\tilde{\gamma}} \tilde{\gamma}/i \leqslant (1/\sqrt{n} + 1)^{n-1} e^{-\sqrt{n}+1} g(\tilde{\gamma}; 2, \sqrt{n}),$$

where $g(\cdot; 2, \sqrt{n})$ denotes the density of a gamma distribution with shape parameter equal to 2 and scale parameter equal to $\sqrt{n}$. Hence it is straightforward to sample $\tilde{\gamma}$ (and hence γ) using a rejection sampler with gamma-distributed proposals. The constant $(1/\sqrt{n} + 1)^{n-1} e^{-\sqrt{n}+1}$

corresponds to the expected number of proposals needed until acceptance. This constant is increasing in i, and as i tends to infinity it tends to $e^{1/2}$. This ensures that the construction of G has complexity $O(n)$.

Given G, we further construct two continuous, piecewise linear bounding functions $\overline{G}$ and $\underline{G}$. Specifically, $\overline{G}$ is a concave function such that $\overline{G}(i) > G(i)$, and $\underline{G}$ is a convex function such that $\underline{G}(i) \leqslant g(i)$. The construction of $\overline{G}$ and $\underline{G}$ has complexity $O(n)$.

3.2. *The image of F*

Regarding the image of F, we observe that in the construction of F, each $Z'_s(x)$ depends on x only through $M'_1(x), \ldots, M'_r(x)$, which in turn depend on x only through $N_1(x), \ldots, N_r(x)$. Hence $F(x)$ depends on x only through the numbers $N_k(x)$, and this implies that the image of F is finite. Thus it is straightforward, at least in principle, to determine the image of F.

The number of valid configurations of $(N_1, \ldots, N_r)$ such that $N_k \geqslant 0$ and $\sum_{k=1}^{r} N_k = n$ is

$$\sum_{k=1}^{r} \binom{n-1}{r-k}\binom{r}{k-1}, \tag{3.4}$$

which gives an upper bound on the cardinality of $F(\Omega)$. If the number is sufficiently low, a perfect sampler would proceed by tracking each point in $F(\Omega)$ through the block. We will return to this idea in § 4.2.

4. *Determining the bounding sets*

We now consider how to construct bounding sets for the update function F described in Section 3. From the construction of F it might appear that we need rather complex bounding sets, such as the specification of a set of possible allocations for each z_s combined with a subset of the simplex S. However, as noted above, F depends on x only through $N_k(x)$. This leads us to consider bounding sets of the form $W = \{x : a_k \leqslant N_k(x) \leqslant b_k\}$. Below we describe a practical way of computing constants a'_k and b'_k such that

$$a_k \leqslant N_k(x) \leqslant b_k \quad \text{implies} \quad a'_k \leqslant N_k(F(x)) \leqslant b'_k \tag{4.1}$$

or, in other words, $F(W) \subseteq W' = \{x : a'_k \leqslant N_k(x) \leqslant b'_k\}$. In particular, by setting $a_k = 0$ and $b_k = n$, we obtain a bounding set that equals Ω.

Suppose that for each data point η_s, we can compute upper and lower bounds $\mathrm{HI}_a^b[s,k]$ and $\mathrm{LO}_a^b[s,k]$ for the component ratios such that

$$\mathrm{LO}_a^b[s,k] \leqslant p_k(\eta_s)M'_k(x) \Big/ \sum_{j=k}^{r} p_j(\eta_s)M'_j(x) \leqslant \mathrm{HI}_a^b[s,k] \tag{4.2}$$

for all $x \in \{x : a_k \leqslant N_k(x) \leqslant b_k\}$. We shall give details of this calculation in § 4.1. Armed with these bounds, consider the variety of possible values the sequential rejection method can assign to $Z'_s(x)$ as x varies over the allowed configurations.

- If $\xi_{s,k} < \mathrm{LO}_a^b[s,k]$, then every single allowed configuration accepts the proposal $Z_s(x) = k$, provided it has not already accepted $Z_s(x) = i$ for $i < k$.
- If $\xi_{s,k} < \mathrm{HI}_a^b[s,k]$, then some (but not necessarily all) configurations accept the proposal $Z_s(x) = k$, so some may still accept a later proposal $Z_s(x) = j$ for $j > k$.
- If $\xi_{s,k} > \mathrm{HI}_a^b[s,k]$, then none of the configurations accept the proposal $Z_s(x) = k$.

We keep track of the possible outcomes for each k by means of the following two sets:

– $\mathrm{LOW}(s) = \{k : \xi_{s,k} < \mathrm{LO}_a^b[s,k]$ and $\xi_{s,i} > \mathrm{HI}_a^b[s,i]$ for all $i < k\}$;
– $\mathrm{HIGH}(s) = \{k : \xi_{s,k} < \mathrm{HI}_a^b[s,k]$ and $\xi_{s,i} > \mathrm{LO}_a^b[s,i]$ for all $i < k\}$.

Obviously, we have $\mathrm{LOW}(s) \subseteq \mathrm{HIGH}(s)$, with the lower set representing the (necessarily unique) component accepted by all allowable configurations (empty if the configurations are split) and the upper set representing all those components which are potentially accepted by at least one allowable configuration. It is now clear that we obtain (4.1) if we choose

$$a'_k = \#\{s : \mathrm{LOW}(s) = \{k\}\} \quad \text{and} \quad b'_k = \#\{s : \mathrm{HIGH}(s) \supseteq \{k\}\}.$$

4.1. *Calculation of* $\mathrm{LO}_a^b[s,k]$ *and* $\mathrm{HI}_a^b[s,k]$

For each $k = 1, \dots, r$, let G_k be an independent realisation of G with corresponding bounding functions $\overline{G}_k$ and $\underline{G}_k$.

As G_k is a non-decreasing function, we have

$$\begin{aligned}\frac{p_k(\eta_s)M'_k(x)}{\sum_{j=k}^r p_j(\eta_s)M'_j(x)} &= \frac{p_k(\eta_s)G_k(x)}{p_k(\eta_s)G_k(x) + \sum_{j=k+1}^r p_j(\eta_s)G_j(x)} \\ &\geqslant \frac{p_k(\eta_s)G_k(a_k+1)}{p_k(\eta_s)G_k(a_k+1) + \max_l \sum_{j=k+1}^r p_j(\eta_s)G_j(l_j+1)},\end{aligned} \tag{4.3}$$

where the maximum is over vectors l belonging to the set

$$\overline{S}(\mathbf{a},\mathbf{b}) := \left\{ l : a_j \leqslant l_j \leqslant b_j \text{ for all } j > k \text{ and } \sum_{j=k+1}^r l_j \leqslant n - \sum_{i \leqslant k} a_i \right\}.$$

The sum in the denominator of (4.3) is bounded as follows:

$$\sum_{j=k+1}^r p_j(\eta_s)G_j(l_j+1) \leqslant \sum_{j=k+1}^r p_j(\eta_s)\overline{G}_j(l_j+1). \tag{4.4}$$

As $\overline{S}(\mathbf{a},\mathbf{b})$ is a convex set and the right-hand side of (4.4) is a convex function, we can maximise the right-hand side of (4.4) using a greedy hill-climbing algorithm. Initially, set $l_j = a_j$ for all $j > k$; then, iteratively allocate the remaining available amount $n - \sum_{j \leqslant k} a_i$ to the component which at each state gives the greatest score increase under the restriction $l \in \overline{S}(\mathbf{a},\mathbf{b})$. This maximisation step has complexity $O(n)$. Accordingly, we set

$$\mathrm{LO}_a^b[s,k] = \frac{p_k(\eta_s)G_k(a_k+1)}{p_k(\eta_s)G_k(a_k+1) + \max_l \sum_{j=k+1}^r p_j(\eta_s)\overline{G}_j(l_j+1)}.$$

Using similar arguments, we obtain

$$\mathrm{HI}_a^b[s,k] = \frac{p_k(\eta_s)G_k(b_k+1)}{p_k(\eta_s)G_k(b_k+1) + \min_l \sum_{j=k+1}^r p_j(\eta_s)\underline{G}_j(l_j+1)},$$

where the minimum is over vectors l belonging to the set

$$\underline{S}(\mathbf{a},\mathbf{b}) := \left\{ l : a_j \leqslant l_j \leqslant b_j \text{ for all } j > k \text{ and } \sum_{j=k+1}^r l_j \geqslant n - \sum_{i \leqslant k} b_i \right\}.$$

The minimum is obtained by means of a greedy hill-descending algorithm. Initially, set $l_j = b_j$ for all $j > k$; then, iteratively reduce the component which at each state gives the greatest score decrease under the restriction $l \in \underline{S}(\mathbf{a},\mathbf{b})$.

In summary, the complexity of calculating $\mathrm{LO}_a^b[s,k]$ and $\mathrm{HI}_a^b[s,k]$ is $O(n)$. As this has to be repeated for all n data points and each component, the total complexity of determining a' and b' is $O(rn^2)$.

We can apply this bounding set to our perfect simulation algorithm as follows. For each C_i we initialise by setting $a_j = 0$ and $b_j = n$ for $j = 1, \dots, r$, which implies $W = \Omega$. Then, sequentially for each $F_{i,k}$ with $k = 1, \dots, K$, update a and b according to the scheme above. If, after the $(K-1)$st update, we have $a_j = b_j$ for $j = 1, \dots, r$, then we declare C_i coalescent; otherwise, we declare C_i non-coalescent. The reason we need $a_j = b_j$ after $K-1$ updates is that $a_j = b_j$

implies coalescence in the allocation space but not necessarily in the weight space. Coalescence in the weight space is obtained after the Kth update.

4.2. *Exact bounding set*

The bounding set obtained by the method above is, in general, not exact. In fact, the simulation study in Section 5 shows that in certain cases the bounding sets thus constructed are so sloppy that coalescence is practically never detected. This leads us to reconsider exact bounds.

Recall that each realisation of F involves generating independent random functions $G_1, \dots, G_r$, where each G_k is generated as in § 3.1 by first generating the skeleton $\{J_k^{(s)}, H_k^{(s)} : s = 1, \dots, L_k\}$. This construction of G implies that the cardinality of the image of F is generally much smaller than (3.4).

To determine the image of F, we first define a basin of attraction: for each $\mathbf{s} = (s_1, \dots, s_r)$ with $s_k \in \{1, \dots, L_k\}$, define

$$\text{Basin}(\mathbf{s}) = \{(N_1, \dots, N_r) : J_k^{(s_k)} - 1 \leqslant N_k \leqslant J_k^{(s_k+1)} - 2\}.$$

Consequently, all states x with $(N_1(x), \dots, N_r(x)) \in \text{Basin}(\mathbf{s})$ are mapped into the same state. The cardinality of the image of F is therefore at most $\prod_{k=1}^{r} L_k$. In practice, the actual number is generally smaller. To see this, observe that Basin$(\mathbf{s})$ only contains valid configurations of N if

$$\sum_{k=1}^{r} (J_k^{(s_k)} - 1) \leqslant n \leqslant \sum_{k=1}^{r} (J_k^{(s_k+1)} - 2),$$

which is not always the case. Furthermore, several basins of attraction may be mapped to the same point in Ω.

Obtaining $F(\Omega)$ involves evaluating F at a number of points, a calculation of order $O(n^r)$. Empirically, it seems that each L_k is of order $\sqrt{n}$, which implies that the number of points would be of order $O(n^{r/2})$. Evaluating F at a single point has complexity $O(nr)$. Hence, given F, the complexity of obtaining $F(\Omega)$ is $O(rn^{(r+1)})$. This should be compared with the complexity $O(rn^2)$ of obtaining the bounds a and b given above. Based on complexity alone, this comparison suggests that using exact bounds is feasible only when the number of components is low.

After determining the exact bounding set $W = F_{i,1}(\Omega)$, a perfect simulation algorithm would simply proceed by tracking each point in W under subsequent random functions $F_{i,2}, \dots, F_{i,K}$. If at the end of the block all points in W have been mapped to the same point, we declare C_i coalescent.

5. *Simulation study*

This section presents the results of a small simulation study of the proposed algorithm. In particular, we investigate the pros and cons of the two ways of constructing bounding sets described in Section 4.

In the following, $\Omega = \{1, \dots, r\}^n \times S$ and $F : \Omega \to \Omega$ denotes the random update function (3.2) described in Section 3. We assume that $C_i = F_{i,K} \circ \dots \circ F_{i,2} \circ F_{i,1}$ where $F_{i,1}, \dots, F_{i,K}$ are K independent realisations of F.

Furthermore, we assume that each π_i is the density of a normally distributed random variable with mean μ_i and common variance σ^2. Below we consider, among other things, different choices of the number of components r and different configurations of the component means $\mu = (\mu_1, \dots, \mu_r)$.

The first part of our simulation study concerns posterior inference for a single data set consisting of $n = 1000$ independent samples from a mixture model with $r = 5$ Gaussian components of mean $\mu = (0, 1, 2, 3, 4)$ and common variance $\sigma^2 = 0.25$. Using the roCFTP

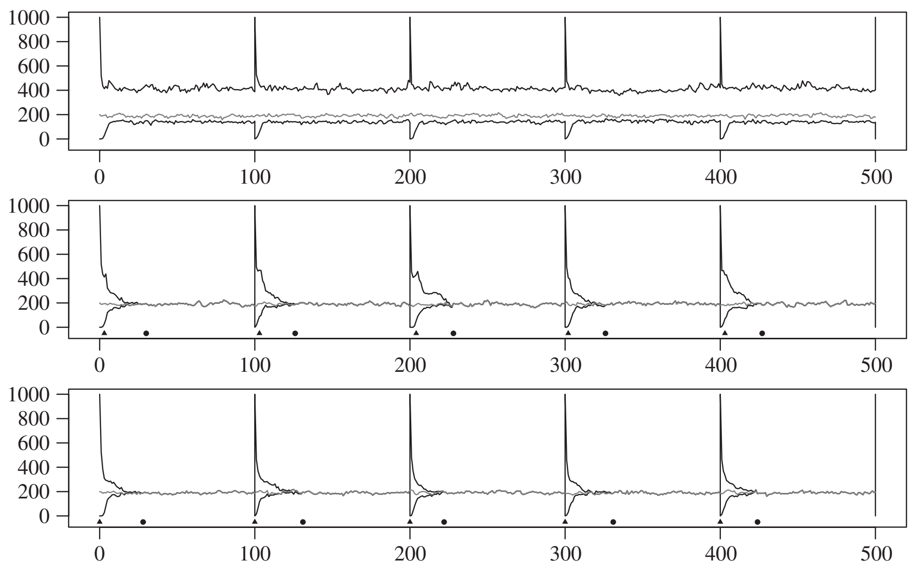

FIGURE 2. *Evolution of the bounds a_1 and b_1 on $N_1(x)$ during five blocks of size $K = 100$, with three different choices of the threshold, for the setup with five components described in Table 1. From top to bottom the threshold is set to be* 0, exp(30) *and* 1001^5. *The grey line corresponds to the target process. A triangle indicates the time at which exact bounding sets start to be used in each block. A circle indicates when coalescence is determined, that is, when $a_k = bK$ for $k = 1, \ldots, r$.*

algorithm described at the end of § 4.1, the top plot of Figure 2 shows how the bounds a_1 and b_1 on $N_1(x)$ evolve during five blocks. As suggested by the plot, declaring C to be coalescent under this algorithm is very unlikely. Increasing the size of the block makes little difference.

If, instead, we use exact bounding sets as described in § 4.2, the result is very different. Notice that the construction of the exact bounding set does not involve the bounds a_k and b_k. Therefore, whenever we use the exact bounding sets, we plot $\min_{x \in W} N_1(x)$ and $\max_{x \in W} N_1(x)$ in place of a_1 and b_1. The bottom plot in Figure 2 shows how these bounds evolve when exact bounding sets are used. Here, the probability of declaring a block of updates coalescent is essentially one. Nevertheless, the 'sloppy' bounding sets do have some merit. It seems reasonable to assume that as close as a_k and b_k can be brought together by these sloppy bounds, they do so in a computationally cheaper fashion than the exact bounds. This suggests combining the two ways of constructing bounding sets.

To combine the two types of bounding sets, we first define the 'volume' $\prod_{k=1}^{r}(b_k - a_k + 1)$ of the bounding set. The basic idea is then to use the non-exact bounding sets until the volume falls below some threshold. After the threshold is reached, we use exact bounding sets for the remainder of that block of updates. This idea is similar to the idea of one-shot coupling introduced in [8], and extends the approach in [2] of tracking a rectangular bounding set until it becomes so small that a more efficient but generally computationally more expensive method is deemed sufficiently cheap at this scale.

As for choosing the threshold, there are two special cases. The first is when the threshold is zero, in which case we never use exact bounding sets. This corresponds to the situation shown in the top panel of Figure 2. The second case is where the threshold is $(n + 1)^r$ or greater, and then we use only exact bounding sets. This corresponds to the bottom panel of Figure 2.

In the middle plot of Figure 2, the threshold is set to $\exp(30) \approx 10^{13}$. This threshold is chosen on the basis of the observed volume of the bounding state when it is in the stable state of the top plot in Figure 2. Comparing this choice of threshold with the case where only exact bounding sets are used, we see that the time until coalescence can be declared is essentially unaffected. More importantly, as can be seen in Table 1, the average CPU time per sample is reduced roughly by two-thirds. Hence, by postponing the use of exact bounding sets until the cheap bounding sets get too sloppy we can speed up the simulation considerably. Table 1 also shows the running times when the block size is only $K = 50$. This has very little impact on the running time compared with the $K = 100$ case. The reason is that when coalescence happens (that is, when $\#W = 1$), the remaining updates have very low computational cost relative to the first update. Based on these observations, one might as well choose K large to ensure a high probability of coalescence.

The second part of our simulation study examines the intuitive expectation that the better-separated the true means are, the easier it should be to estimate the unknown weights. Table 2 compares perfect simulation of posterior weights in two situations with three components ($r = 3$) that essentially differ only in how well-separated the mixture components are. In both cases, it appears that using only exact bounds yields the best results. This is explained by the fact that the complexity of the two types of bounds is very similar when the number of components is this low. When the computational costs are similar, the exact bounds are obviously the better choice. Looking at the block size K, we see that, not surprisingly, the optimum block size increases as the component means become less separated.

5.1. *Generalisation: Gaussian mixtures with unknown means*

So far, we have described how to generate perfect samples from the posterior weight distribution when the mixture components are known. It is possible to extend our approach to the case where the components are specified by unknown parameters. Specifically, we consider the case where the r components are Gaussian with common, known variance σ^2 and individual, unknown means $\mu_1, \ldots, \mu_r$. As for the Bayesian setup, we assume a uniform prior on the weights and a normal prior for each μ_j. Unfortunately, in practice this extension of our algorithm seems to be feasible only for fairly small data sets ($n < 5$) where the data consist

TABLE 1. *Simulation study based on inference for a single data set generated using $\mu = (0, 1, 2, 3, 4)$ and $n = 1000$. For the algorithm, we have used blocks of size $K = 50$ and $K = 100$ and three different choices of the threshold (Thl). In each entry of the table, the first number shows the average CPU time per sample, and the number in brackets shows the estimated probability of a block being declared coalescent. Both CPU time and probability of coalescence are based on running the algorithm until* 100 *samples have been generated. 'NA' means that no samples were produced within a reasonable time.*

	Thl		
K	0	exp(30)	1001^5
50	NA (NA)	72.6 (0.99)	209.3 (1.00)
100	NA (NA)	72.8 (1.00)	217.0 (1.00)

TABLE 2. *Like Table 1, except here we compare the inferences for two data sets generated under different setups. For both data sets the sample size is $n = 1000$, but in the left table $\mu = (0, 1, 2)$ while in the right table $\mu = (0, 0.5, 1)$.*

	Thl	
K	0	1001^3
150	1.58 (1.00)	0.53 (1.00)
100	1.52 (1.00)	0.50 (1.00)
50	1.48 (1.00)	0.50 (1.00)
25	1.92 (0.75)	0.50 (0.98)

	Thl	
K	exp(20)	1001^3
150	1.02 (0.94)	0.80 (0.94)
100	1.41 (0.65)	1.21 (0.59)
50	NA (1.00)	NA (NA)
25	NA (1.00)	NA (NA)

of one cluster. In other words, the algorithm works best if the likelihood that the data comes from just one Gaussian distribution is high. For more details on this, see [1].

6. *Hidden Markov models*

In this section we extend our approach to perfect simulation of the posterior transition probabilities in a discrete-state hidden Markov model. This essentially means that instead of assuming that the allocation variables $Z_0, \dots, Z_n$ are independent and identically distributed, we assume that $Z = (Z_0, \dots, Z_n)$ is a Markov chain. For simplicity, we assume that Z is a two-state Markov chain with transition matrix $Q = \{q_{ij} : i, j \in \{1, 2\}\}$; that is, $\mathbb{P}(Z_{s+1} = j \mid Z_s = i) = q_{ij}$. This implies a state vector $x = (z_0, \dots, z_n, q_{11}, q_{12}, q_{21}, q_{22})$.

It is easily seen that this Markov chain has stationary distribution with density proportional to $(1 - q_{22}, 1 - q_{11}) = (q_{21}, q_{12})$. As before, we assume a priori that $\eta_1, \dots, \eta_n$ are independent given Z and that $\eta_s | Z_s = i \sim p_i$. Assuming that the Markov chain Z is started in equilibrium, the joint likelihood for $\eta = (\eta_0, \dots, \eta_n)$ and Z given Q is

$$\pi(\eta, Z \mid Q) = \prod_{s=1}^{n} q_{Z_{s-1} Z_s} \prod_{s=0}^{n} p_{Z_s}(\eta_s)(q_{21} + \mathbb{1}[Z_0 = 2](q_{12} - q_{21}))/(q_{21} + q_{12}).$$

To obtain a simple posterior, we assume that the joint prior density for q_{11} and q_{22} is proportional to $q_{12} + q_{21}$. The posterior density is then proportional to

$$\prod_{s=0}^{n} p_{z_s}(\eta_s) \prod_{s=1}^{n} q_{z_{s-1} z_s}(q_{21} + \mathbb{1}[z_0 = 2](q_{12} - q_{21})).$$

6.1. *Specifying the update function*

The posterior can be sampled using a Gibbs sampler, upon noticing that conditional on Z and η the posterior distributions of q_{11} and q_{22} are independent and beta:

$$q_{11} | Z = z, \eta \sim \text{beta}(N_{11}(x) + 1, N_{12}(x) + \mathbb{1}[z_0 = 2] + 1) \tag{6.1}$$

and

$$q_{22} | Z = z, \eta \sim \text{beta}(N_{22}(x) + 1, N_{21}(x) + \mathbb{1}[z_0 = 1] + 1),$$

where

$$N_{ij}(x) = \#\{s : 1 \leqslant s \leqslant n, z_{s-1} = i, z_s = j\}$$

is the number of transitions from state i to state j in z. The conditional posterior probabilities for the allocation variables are

$$\begin{aligned}
\mathbb{P}(Z_0 = 1 \mid Q, \eta, Z_{-0}) &= p_1(\eta_0) q_{21} q_{1 z_1} / [p_1(\eta_0) q_{21} q_{1 z_1} + p_2(\eta_0) q_{12} q_{2 z_1}], \\
\mathbb{P}(Z_s = 1 \mid Q, \eta, Z_{-s}) &= p_1(\eta_s) q_{z_{s-1} 1} q_{1 z_{s+1}} / [p_1(\eta_s) q_{z_{s-1} 1} q_{1 z_{s+1}} + p_2(\eta_s) q_{z_{s-1} 2} q_{2 z_{s+1}}] \\
&\qquad \text{for } 1 \leqslant s \leqslant n - 1, \\
\mathbb{P}(Z_n = 1 \mid Q, \eta, Z_{-n}) &= p_1(\eta_n) q_{z_{n-1} 1} / [p_1(\eta_n) q_{z_{n-1} 1} + p_2(\eta_n) q_{z_{n-1} 2}],
\end{aligned}$$

where $Z_{-s} = (Z_0, \dots, Z_{s-1}, Z_{s+1}, \dots, Z_n)$.

As in Section 3, the Gibbs sampler can be expressed as a random update function

$$F(x) = (Z'_0(x), \dots, Z'_n(x), q'_{11}(x), q'_{22}(x)).$$

To generate $q'_{11}(x)$ from the beta distribution (6.1), we generate gamma-distributed variables $G_{11} \sim \Gamma(N_{11}(x) + 1)$ and $G_{12}(x) \sim \Gamma(N_{12}(x) + \mathbb{1}[Z_0 = 2] + 1)$ and then set $q'_{11}(x) = G_{11}(x)/(G_{11}(x) + G_{12}(x))$. The variable $q'_{22}(x)$ is generated in a similar way. The gamma-distributed random variables are generated as in Section 3; that is, each G_{ij}, $i, j \in \{1, 2\}$, is an independent realisation of the random function G described in § 3.1. Then

$$q'_{11}(x) = G_{11}(N_{11}(x)+1)/[G_{11}(N_{11}(x)+1) + G_{12}(N_{12}(x) + \mathbb{1}[Z_0 = 2] + 1)],$$

and $q'_{22}(x)$ is given in a similar way.

To generate $Z'_0(x)$, first generate $\xi_0 \sim \mathrm{U}[0,1]$ and then, if

$$\xi_0 \leqslant p_1(\eta_0)q'_{21}q'_{1z_1}/[p_1(\eta_0)q'_{21}q'_{1z_1} + p_2(\eta_0)q'_{12}q'_{2z_1}],$$

set $Z'_0 = 1$; otherwise, set $Z'_0(x) = 2$.

To generate $Z'_s(x)$ for $s = 1, \dots, n-1$, first generate $\xi_s \sim \mathrm{U}[0,1]$ and then, if

$$\xi_s \leqslant p_1(\eta_s)q'_{z'_{s-1}}q'_{1z_{t+1}}/[p_1(\eta_s)q'_{z'_{s-1}1}q'_{1z_{s+1}} + p_2(\eta_s)q'_{z'_{s-1}2}q'_{2z_{s+1}}],$$

set $Z'_s(x) = 1$; otherwise, set $Z'_s(x) = 2$.

To generate $Z'_n(x)$, first generate $\xi_n \sim \mathrm{U}[0,1]$ and then, if

$$\xi_n \leqslant p_1(\eta_n)q'_{z'_{n-1}1}/[p_1(\eta_n)q'_{z'_{n-1}1} + p_2(\eta_n)q'_{z'_{n-1}2}],$$

set $Z'_n(x) = 1$; otherwise, set $Z'_n(x) = 2$.

6.2. *Determining the bounding sets*

Unlike in Section 4, here $F(x)$ depends on the exact configuration of x. Consequently, we consider bounding sets of the form $W[A] = \{x : z_i \in A_i\}$, where $A = (A_1, \dots, A_n)$ and $A_1, \dots, A_n$ are non-empty subsets of $\{1,2\}$. Given A and a realisation of F, we give a practical way of determining $A' = (A'_1, \dots, A'_n)$ so that $x \in W[A]$ implies $F(x) \in W[A']$.

Assume that for each s we can find bounds $\mathrm{LO}_s[A]$ and $\mathrm{HI}_s[A]$ such that

$$\mathrm{LO}_s[A] \leqslant \mathbb{P}(Z_s(x) = 1 \mid Q(x), Z_{-s}(x), \eta) \leqslant \mathrm{HI}_s[A] \quad \text{for all } x \in W[A],$$

where

$$Z_{-s}(x) = (Z'_0(x), \dots, Z'_{s-1}(x), z_{s+1}, \dots, z_n).$$

Then, if $\xi_s \leqslant \mathrm{LO}[A]_s$, we know that for any $x \in W[A]$ we have $Z'_s(x) = 1$, hence we set $A'_s = \{1\}$. If $\mathrm{LO}[A]_s < \xi_s \leqslant \mathrm{HI}[A]_s$, then the state of $Z'_s(x)$ may be either 1 or 2 depending on x, hence $A'_s = \{1,2\}$. Finally, if $\xi_s > \mathrm{HI}[A]_s$, then $Z'_s(x) = 2$ for all $x \in W[A]$ and hence $A'_s = \{2\}$.

When finding the bounds, we consider three separate cases: $s = 0$, $1 \leqslant s \leqslant n-1$ and $s = n$. In all three cases, finding the bounds involves maximisation or minimisation over a (super)set of possible configurations of $N = (N_{11}, N_{12}, N_{21}, N_{22})$. To determine this set, we notice that for the two-state Markov chain considered here, it is clear that the difference between N_{12} and N_{21} is at most one. Further, as $\sum_{i=1}^{2}\sum_{j=1}^{2} N_{ij} = n$, this implies

$$N_{12} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[z_0 = 1]((n - N_{11} - N_{22}) \bmod 2)$$

and

$$N_{21} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[z_0 = 2]((n - N_{11} - N_{22}) \bmod 2).$$

Assume we have bounds $\underline{N}_{ij}$ and $\overline{N}_{ij}$ such that $\underline{N}_{ij} \leqslant N_{ij}(x) \leqslant \overline{N}_{ij}$ for all $x \in W[A]$ and $i, j = 1, 2$. Then $\underline{N}_{11} \leqslant N_{11} \leqslant \overline{N}_{11}$, and given N_{11}, we have $\underline{N}_{22} \leqslant N_{22} \leqslant \min(\overline{N}_{22}, n - N_{11})$. Given N_{11} and N_{22}, for each value of $\tilde{z} \in A_0$ we have

$$N_{12} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[\tilde{z} = 1]((n - N_{11} - N_{22}) \bmod 2)$$

and

$$N_{21} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[\tilde{z} = 2]((n - N_{11} - N_{22}) \bmod 2).$$

This leads us to define the set

$$\begin{aligned}N[A] = \{(N_{11}, N_{12}, N_{21}, N_{22}) : \underline{N}_{11} \leqslant N_{11} \leqslant \overline{N}_{11},\ \underline{N}_{22} \leqslant N_{22} \leqslant \min(n - N_{11}, \overline{N}_{22}),\\ N_{12} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[\tilde{z} = 1]((n - N_{11} - N_{22}) \bmod 2) \text{ and}\\ N_{21} = \lfloor (n - N_{11} - N_{22})/2 \rfloor + \mathbb{1}[\tilde{z} = 2]((n - N_{11} - N_{22}) \bmod 2) \text{ for } \tilde{z} \in A_0\}.\end{aligned}$$

We are now ready to define the bounds. In the $s = 0$ case,

$$\mathrm{LO}_0[A] = \min_{\substack{N \in N[A]\\ \tilde{z} \in A_1}} \frac{p_1(\eta_0)(1 - q'_{22}(N))q_{1\tilde{z}_1}(N)}{p_1(\eta_0)(1 - q'_{22}(N))q'_{1\tilde{z}_1} + p_2(\eta_0)(1 - q'_{11}(N))q'_{2\tilde{z}_1}(N)}$$

where q'_{11} and q'_{22} are defined as in § 6.1, $q'_{12}(N) = 1 - q'_{11}(N)$ and $q'_{21}(N) = 1 - q'_{22}(N)$.

For $1 \leqslant s \leqslant n - 1$,

$$\mathrm{LO}_s[A] = \min_{\substack{N \in N[A]\\ \tilde{z} \in A'_{s-1}, \tilde{\tilde{z}} \in A_{t+1}}} \frac{p_1(\eta_s)q'_{\tilde{z}1}(N)q'_{1\tilde{\tilde{z}}}(N)}{p_1(\eta_s)q'_{\tilde{z}1}(N)q'_{1\tilde{\tilde{z}}}(N) + p_2(\eta_s)q'_{\tilde{z}2}(N)q'_{2\tilde{\tilde{z}}}(N)}; \tag{6.2}$$

and for $s = n$,

$$\mathrm{LO}_n[A] = \min_{\substack{N \in N[A]\\ \tilde{z} \in A'_{n-1}}} \frac{p_1(\eta_n)q'_{\tilde{z}1}(N)}{p_1(\eta_n)q'_{\tilde{z}1}(N) + p_2(\eta_n)q'_{\tilde{z}2}(N)}. \tag{6.3}$$

Each HI_s is defined in exactly the same way as LO_s except that minimisation is replaced by maximisation. Notice that (6.2) and (6.3) involve the set A'_{s-1}, which implies that LO_{s-1} and HI_{s-1} have to be determined before LO_s and HI_s can be determined.

The minimisation and maximisation over $N[A]$ has a worst-case complexity of $O(n^2)$. With n data points, the worst-case complexity for determining A' is $O(n^3)$.

6.3. *Generating perfect samples*

As in Section 5 we use Wilson's read-once algorithm for perfect simulation, but unlike in Section 5 we do not consider exact bounding sets since they are computationally infeasible in the present setting.

Table 3 shows the computational costs for two data sets of different sizes. In both cases, p_i is the density of a normally distributed random variable with mean μ_i and variance σ^2. Furthermore, each C_i is a compound of ten independent update functions. It appears that a quadrupling of the size of the data set results in a dramatic increase in the computational cost, as would be expected given the complexity of the algorithm.

7. *Discussion*

We have introduced a general method for perfect simulation applicable to mixture models. Although the applicability of our techniques does not extend as far as the class of problems addressed by MCMC, the methods are exact and more widely relevant than existing methods for perfect simulation.

TABLE 3. *Computational costs for data sets of different sizes. Each row corresponds to a single data set. Each data set is a realisation of the hidden Markov model specified by the parameters listed in the first six columns. The perfect simulation algorithm was run until it had produced 100 perfect samples. The time taken to produce the 100 perfect samples is shown in the last column.*

n	μ_1	μ_2	q_{11}	q_{22}	σ	CPU time (seconds)
25	−1	1	0.3	0.6	0.5	18
100	−1	1	0.3	0.6	0.5	102

Acknowledgement. We would like to thank an anonymous referee for many helpful comments that have helped to improve the paper.

Appendix A. *Validity of the algorithm in* § 3.1

We now show that the random function G presented in § 3.1 has the property that $G(i) \sim \Gamma(i)$. The construction of G involves constructing a random number of pairs (γ, u) of random variables. Let $S = \{(\gamma_1, u_1), (\gamma_2, u_2), \ldots, (\gamma_L, u_L)\}$ denote the sequence of these pairs of random variables. Define the set $g_i = \{(\gamma, u) : 0 \leqslant u \leqslant g(\gamma; i)\}$. By construction, each set $g_i \cap S$ contains exactly one point from S; let $(\tilde{\gamma}_i, \tilde{u}_i)$ be the unique point in $g_i \cap S$. We now show by induction that $(\tilde{\gamma}_i, \tilde{u}_i)$ is uniformly distributed on g_i for all $i = 1, 2, \ldots, n+1$. By construction, (γ_1, u_1) is uniformly distributed on g_1 and hence so is $(\tilde{\gamma}_1, \tilde{u}_1)$. Assume, for $i = 1, \ldots, \kappa - 1$, that $(\tilde{\gamma}_i, \tilde{u}_i)$ is uniformly distributed on g_i. Notice that $g_\kappa = (g_{\kappa-1} \cap g_\kappa) \cup (g_\kappa \backslash g_{\kappa-1})$ and $(g_{\kappa-1} \cap q_\kappa) \cap (g_\kappa \backslash g_{\kappa-1}) = \emptyset$. By assumption, with probability $|g_{\kappa-1} \cap g_\kappa|$ we have $(\tilde{\gamma}_{\kappa-1}, \tilde{u}_{\kappa-1}) \in g_{\kappa-1} \cap g_\kappa$, and hence $(\tilde{\gamma}_\kappa, \tilde{u}_\kappa) = (\tilde{\gamma}_{\kappa-1}, \tilde{y}_{\kappa-1})$. In other words, with probability $|g_{\kappa-1} \cap g_\kappa|$ the point $(\tilde{\gamma}_\kappa, \tilde{u}_\kappa)$ is uniformly distributed on $g_{\kappa-1} \cap g_\kappa$. With probability $|g_{\kappa-1} \backslash g_\kappa| = 1 - |g_{\kappa-1} \cap g_\kappa|$ we have $(\tilde{\gamma}_{\kappa-1}, \tilde{u}_{\kappa-1}) \in g_{\kappa-1} \backslash g_\kappa$, that is, $(\tilde{\gamma}_{\kappa-1}, \tilde{u}_{\kappa-1}) \notin g_\kappa$. In this case the algorithm generates a new pair (γ, u) uniformly on $g_\kappa \backslash g_{\kappa-1}$; that is, with probability $|g_{\kappa-1} \backslash g_\kappa|$ the point $(\tilde{\gamma}_\kappa, \tilde{u}_\kappa)$ is uniformly distributed on $g_\kappa \backslash g_{\kappa-1}$. Since $|g_\kappa \backslash g_{\kappa-1}| = |g_{\kappa-1} \backslash g_\kappa|$, we are done.

References

1. K. K. Berthelsen, L. A. Breyer and G. O. Roberts, 'Perfect posterior simulation for mixture and hidden Markov models', Technical Report 07-23, Centre for Research in Statistical Methodology (CRiSM), University of Warwick, 2007.
2. A. Beskos and G. O. Roberts, 'One-shot CFTP; application to a class of truncated Gaussian densities', *Methodol. Comput. Appl. Probab.* 31 (2005) 407–437.
3. L. A. Breyer and G. O. Roberts, 'Catalytic perfect simulation', *Methodol. Comput. Appl. Probab.* 3 (2001) 161–177.
4. G. Casella, K. L. Mengersen, C. P. Robert and D. M. Titterington, 'Perfect samplers for mixtures of distributions', *J. Roy. Statist. Soc. Ser. B* 64 (2002) 777–790.
5. J. Diebolt and C. P. Robert, 'Estimation of finite mixture distributions through Bayesian sampling', *J. Roy. Statist. Soc. Ser. B* 56 (1994) 363–375.
6. J. P. Hobert, C. P. Robert and D. M. Titterington, 'On perfect simulation for some mixtures of distributions', *Stat. Comput.* 9 (1999) 223–252.
7. J. G. Propp and D. B. Wilson, 'Exact sampling with coupled Markov chains and applications to statistical mechanics', *Random Structures Algorithms* 9 (1996) 223–252.
8. G. O. Roberts and J. S. Rosenthal, 'One-shot coupling for certain stochastic recursive sequences', *Stochastic Process. Appl.* 99 (2002) 195–208.
9. D. B. Wilson, 'How to couple from the past using a read-once source of randomness', *Random Structures Algorithms* 16 (2000) 85–113.

Kasper K. Berthelsen
Department of Mathematical Sciences
Aalborg University
9220 Aalborg Øst
Denmark

kkb@math.aau.dk

Laird A. Breyer
Department of Statistics
Lancaster University
Bailrigg
Lancaster LA1 4YF
United Kingdom

http://www.lbreyer.com/

Gareth O. Roberts
Department of Statistics
University of Warwick
Coventry CV4 7AL
United Kingdom

gareth.o.roberts@warwick.ac.uk