Real-Parameter Evolutionary Monte Carlo With Applications to Bayesian Mixture Models

Faming LIANG and Wing Hung WONG

We propose an evolutionary Monte Carlo algorithm to sample from a target distribution with real-valued parameters. The attractive features of the algorithm include the ability to learn from the samples obtained in previous steps and the ability to improve the mixing of a system by sampling along a temperature ladder. The effectiveness of the algorithm is examined through three multimodal examples and Bayesian neural networks. The numerical results confirm that the real-coded evolutionary algorithm is a promising general approach for simulation and optimization.

KEY WORDS: Crossover; Evolutionary Monte Carlo; Exchange; Genetic algorithm; Markov chain Monte Carlo; Metropolis algorithm; Mixture model; Mutation; Neural network; Parallel tempering.

1. INTRODUCTION

Markov chain Monte Carlo (MCMC) methods have played an important role in the development of science and technology. This method originated in statistical physics (Hastings 1970; Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953) and in recent years has been widely used in Bayesian statistical computations. (For a recent overview, see Chen, Shao, and Ibrahim 2000.) Several general promising ideas have been found in the design of effective Monte Carlo methods. The idea of a temperature ladder has been used in parallel tempering (Geyer 1991), simulated tempering (Geyer and Thompson 1995; Marinari and Parisi 1992), exchange Monte Carlo (Hukushima and Nemoto 1996), and dynamic weighting (Wong and Liang 1997). This idea works by simulating a sequence of distributions along a temperature ladder. Simulation at the high temperature levels help the system overcome barriers of the energy landscape. The methods have been applied successfully in the simulation of spin-glass models (Hukushima and Nemoto 1996; Liang and Wong 1999; Marinari and Parisi 1992) and statistical inference (Geyer and Thompson 1995).

The idea of importance sampling has been used to design new MCMC methods. In the multicanonical method (Berg and Neuhaus 1991), the MCMC method is designed to sample a microcanonical distribution, and importance weights are used to obtain the correct expectation. In dynamic weighting (Wong and Liang 1997), the importance weight itself becomes a dynamic variable and helps the system overcome the high energy barriers.

The third general idea is the use of population. Parallel tempering (Geyer 1991) can be viewed as a semi-population method. It evolves a population of samples, with each sample of the population updated individually. An early example of a true population method is adaptive direction sampling (Gilks, Roberts, and George 1994), in which a population of samples is simulated in parallel; at each iteration, one sample is randomly selected from the current population to update by a line sampling step performed along a direction pointing to another sample randomly selected from the current population. This method was improved by Liu, Liang, and Wong (2000), who suggested that the line sampling be performed along a direction pointing to a local mode found by a local optimization procedure initialized at another sample randomly selected from the current population. Local optimization can be accomplished by a few steps of conjugate gradient iterations, and thus the method is called *conjugate gradient Monte Carlo* (CGMC).

Recently, a population and temperature ladder based new Monte Carlo method, the so-called *evolutionary Monte Carlo* (EMC) method (Liang and Wong 2000), was proposed to sample from a distribution defined on a space of finite binary sequence. This method works by simulating a population of samples in parallel, with a different temperature attached to each sample. The population is updated by mutation, crossover and exchange operations with a high similarity with a genetic algorithm (Goldberg 1989; Holland 1975). EMC has the learning ability of the genetic algorithm as well as the fast mixing ability of parallel tempering (simulated tempering).

But EMC encounters the same difficulty as genetic algorithms in applications to real parameter problems; namely, the usual crossover operators for binary-coded chromosomes are not as effective in the case that the chromosomes are coded by real parameters. If one starts with a population, then the crossover operator allows one to reach only a finite number of points in the parameter space—those points whose parameter components are selected from the corresponding components of population members. In recent years, several crossover operators have been proposed for the real-coded chromosomes in the context of genetic algorithms, including linear crossover (Wright 1991), BLX-α crossover (Eshelman and Schaffer 1993), and UNDX crossover (Ono and Kobayashi 1997). All of these operators need some modifications for applications in MCMC to satisfy the invariance or reversibility property of the Markov chain transitions, however.

In this article we extend EMC to sample from a distribution defined on a real space. We propose a snooker crossover operator for the real-coded chromosomes in the framework of MCMC. We illustrate the algorithm through three multimodal

Faming Liang is Assistant Professor, Department of Statistics & Applied Probability, National University of Singapore, Singapore 117543 (E-mail: *stalfm@math.nus.edu.sg*). Wing Hung Wong is Professor, Department of Biostatistics, Harvard School of Public Health, Boston, MA 02115 (E-mail: *wwong@hsph.harvard.edu*). Liang's research was supported in part by Academic Research grant RP-3992648; Wong's research was supported in part by National Science Foundation grant DMS-9977096. The authors thank professor Martin A. Tanner for suggesting the galaxy data example.

© 2001 American Statistical Association
Journal of the American Statistical Association
June 2001, Vol. 96, No. 454, Theory and Methods

examples and Bayesian neural networks. Numerical studies demonstrate that the real-coded EMC algorithm offers a substantial improvement over parallel tempering in simulation and optimization.

The article is organized as follows. The binary-coded EMC algorithm is briefly reviewed in Section 2, and the real-coded EMC algorithm and the snooker crossover operator are described in Section 3. The use of the real-coded EMC algorithm is illustrated through three examples in Section 4. The applications of the algorithm in Bayesian neural networks are given in Section 5. A brief discussion in Section 6 concludes the article.

2. EVOLUTIONARY MONTE CARLO

Here we briefly review EMC (Liang and Wong 2000). Suppose that we want to sample from a distribution defined on a space of finite binary sequence,

$$f(x) \propto \exp\{-H(x)/\tau\}, \tag{1}$$

where x is a d-dimensional binary vector $x = (\beta_1, \beta_2, \ldots, \beta_d)$ with $\beta_i \in \{0, 1\}$ and τ is a scale parameter (a so-called temperature, which can be any value in which we are interested); $H(x)$ is called a "fitness" function in terms of genetic algorithms. In general, x is often a sample from a high-dimensional space and $H(x)$ is the negative of the log-density (up to an additive constant) of x.

First, a sequence of distributions $f_1(x), \ldots, f_N(x)$ is constructed as follows:

$$f_i(x) = \frac{1}{Z(t_i)} \exp\{-H(x)/t_i\},$$

for $i = 1, \ldots, N$, where $Z(t_i) = \sum_{x_i} \exp\{-H(x_i)/t_i\}$. The temperatures $t_1, \ldots, t_N$ form a ladder with the ordering $t_1 > \cdots > t_N$. For convenience, we denote the ladder by $\boldsymbol{t} = \{t_1, \ldots, t_N\}$. Note that we always set $t_N = \tau$ and that $f_N(x) = f(x)$ corresponds to the target distribution from which to sample. Let $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ denote a population of samples, where x_i is a sample from $f_i(x)$ and is called a chromosome or an individual in terms of the genetic algorithm, and N is called a population size. In EMC, the Markov chain state is augmented as the population $\boldsymbol{x}$ instead of a single sample x_i, and the Boltzmann distribution of the population is

$$f(\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{t})} \exp\left\{ -\sum_{i=1}^{N} H(x_i)/t_i \right\}, \tag{2}$$

where $Z(\boldsymbol{t}) = \prod_{i=1}^{N} Z(t_i)$. The population is updated by mutation, crossover, and exchange operators (described later).

In the mutation operator, a chromosome, say x_k, is uniformly chosen from the current population $\boldsymbol{x} = \{x_1, \ldots, x_k, \ldots, x_N\}$, then mutated to a new chromosome y_k by reversing the values of some bits that are also chosen randomly. A new population is proposed as $\boldsymbol{y} = \{x_1, \ldots, y_k, \ldots, x_N\}$, and it is accepted or rejected according to the Metropolis–Hastings rule.

In the crossover operator, one chromosome pair, say x_i and x_j, is selected from the current population $\boldsymbol{x}$ according to some selection procedure, such as random selection or roulette wheel selection. In random selection, a chromosome is uniformly selected from the current population. In roulette wheel selection, a chromosome is selected with a probability proportional its Boltzmann weight $w(x_i) \propto \exp(-H(x_i)/t_s)$, where t_s is a selection temperature that takes a value the same as or close to t_N. Two new "offspring," y_i and y_j, are generated by one-point, k-point, uniform, or adaptive crossovers. A new population is proposed as $\boldsymbol{y} = \{x_1, \ldots, y_i, \ldots, y_j, \ldots, x_N\}$, and it is accepted or rejected according to the Metropolis–Hastings rule. In the one-point crossover, y_i and y_j are generated as follows. First, an integer crossover point is drawn uniformly on $\{1, 2, \ldots, d\}$; then y_i and y_j are constructed by swapping the gene to the right of the crossover point between parents. The following graph illustrates the one-point crossover operator:

$$\begin{array}{cc} (\beta_1^i, \ldots, \beta_d^i) & (\beta_1^i, \ldots, \beta_c^i, \beta_{c+1}^j, \ldots, \beta_d^j) \\ & \Longrightarrow \\ (\beta_1^j, \ldots, \beta_d^j) & (\beta_1^j, \ldots, \beta_c^j, \beta_{c+1}^i, \ldots, \beta_d^i), \end{array}$$

where c is called a crossover point. If there are k crossover points, then it is called a k-point crossover. One extreme case is the uniform crossover, in which the value of each position (i.e., the genotype) of y_i is randomly chosen from the two parental genotypes and the corresponding genotype of y_j is assigned to the parental genotype not chosen by y_i. The adaptive crossover is more complicated, (see Liang and Wong 2000 for details).

The exchange operator is similar to that used in parallel tempering and EMC.

3. REAL-CODED EVOLUTIONARY MONTE CARLO

Now we extend EMC to sample from a distribution defined on a real space with real-coded chromosomes, that is, $x = \{\beta_1, \beta_2, \ldots, \beta_d\}$ with $\beta_i \in R$. We define the corresponding mutation, crossover, and exchange operators.

3.1 Mutation

We define the mutation operator as an additive Metropolis–Hastings move. One chromosome, say x_k, is uniformly chosen from the current population $\boldsymbol{x}$. A new chromosome is generated by adding a random vector e_k so that

$$y_k = x_k + e_k, \tag{3}$$

where e_k is usually chosen for the mutation operation to have a moderate acceptance probability, (e.g., .2–.5), as suggested by Gelman, Roberts, and Gilks (1996). The new population $\boldsymbol{y} = \{x_1, \ldots, y_k, \ldots, x_N\}$ is accepted with probability $\min(1, r_m)$ according to the Metropolis–Hastings rule,

$$\begin{aligned} r_m &= \frac{f(\boldsymbol{y})}{f(\boldsymbol{x})} \frac{T(\boldsymbol{x}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x})} \\ &= \exp\{-(H(y_k) - H(x_k))/t_k\} \frac{T(\boldsymbol{x}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x})}, \end{aligned} \tag{4}$$

where $T(\cdot|\cdot)$ denotes the transition probability between populations.

3.2 Crossover

One chromosome pair, say x_i and x_j ($i \neq j$), is selected from the current population $\boldsymbol{x}$ through roulette wheel or random selection. Two "offspring," y_i and y_j, are generated according to some crossover operator (described later). A new population $\boldsymbol{y}$ is proposed as $\boldsymbol{y} = \{x_1, \ldots, y_i, \ldots, y_j, \ldots, x_N\}$, and it is accepted with probability $\min(1, r_c)$ according to the Metropolis–Hastings rule,

$$r_c = \frac{f(\boldsymbol{y})}{f(\boldsymbol{x})}\frac{T(\boldsymbol{x}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x})} = \exp\Big\{-(H(y_i) - H(x_i))/t_i - (H(y_j) - H(x_j))/t_j\Big\}\frac{T(\boldsymbol{x}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x})}, \quad (5)$$

where $T(\boldsymbol{y}|\boldsymbol{x}) = P((x_i, x_j)|\boldsymbol{x})P((y_i, y_j)|(x_i, x_j))$, $P((x_i, x_j)|\boldsymbol{x})$ denotes the selection probability of (x_i, x_j) from the population $\boldsymbol{x}$, and $P((y_i, y_j)|(x_i, x_j))$ denotes the generating probability of (y_i, y_j) from the parental chromosomes (x_i, x_j).

3.2.1 Real Crossover. One type of crossover operator used in this article is the so called "real crossover," which includes one-point, k-point, and uniform crossover operators. They are the same as that used in EMC. Wright (1991) called this the "real crossover" to indicate that they are applied to the real-coded chromosomes. One feature common to all of them is symmetry; that is, $P((y_i, y_j)|(x_i, x_j)) = P((x_i, x_j)|(y_i, y_j))$.

3.2.2 Snooker Crossover. The snooker crossover operator is based on the snooker algorithm (Gilks et al. 1994; Roberts and Gilks 1994). In the snooker algorithm, a population of iid samples is simulated in parallel. At each iteration, one sample, called a *current point*, is selected uniformly from the current population, and is then updated by a line sampling step performed along a direction passing through another point, called an *anchor point*. The choice of the anchor point may depend on any or all of the samples in the current population except the current point. If the update leaves the conditional distribution on the line invariant, then the resulting new point is also an iid sample of the current set.

In this article, we generalize this operator to the case where the samples of the current set are not necessarily identically distributed. The snooker crossover operator proceeds as follows:

1. Uniformly select one chromosome, say x_i, from the current population $\boldsymbol{x}$.
2. Select the other chromosome, say x_j, from the subpopulation $\boldsymbol{x} \setminus \{x_i\}$ with a probability proportional to its Boltzmann weight $w_j = \exp\{-H(x_j)/t_s\}$, where t_s is called a selection temperature.
3. Let $\mathbf{e} = (x_j - x_i)/\|x_j - x_i\|$, and $y_i = x_j + r\mathbf{e}$, where $r \in (-\infty, \infty)$ is a random variable sampled from the density

$$f(r) \propto |r|^{d-1}\pi(x_j + r\mathbf{e}). \quad (6)$$

4. Construct a new population by replacing x_i with the "offspring" y_i, and replace $\boldsymbol{x}$ by $\boldsymbol{y}$.

Note that the line sampling step can be replaced by one or several Metropolis–Hastings moves or the Griddy-Gibbs sampler (Ritter and Tanner 1992) in the case where sampling directly from $f(r)$ is difficult. The following theorem shows that the snooker crossover operator is indeed a proper transition, which leaves the Boltzmann distribution (2) invariant. To prove the theorem, we first introduce a lemma, which is a generalized version of lemma 3.1 of Roberts and Gilks (1994) and was proven by Liu et al. (2000).

Lemma 1. Suppose that $x \sim \pi$ and z is any fixed point in a d-dimensional space. Let $\mathbf{e} = (x - z)/\|x - z\|$ be a unit vector. If r is drawn from its conditional distribution on the direction e, [i.e., $f(r) \propto |r|^{d-1}\pi(z + r\mathbf{e})$], then $y = z + r\mathbf{e}$ follows distribution π. If z is generated from a distribution independent of x, then y is independent of z and has density $\pi(y)$.

Theorem 1. Under the setting of the real-coded EMC algorithm, the Boltzmann distribution (2) is invariant with respect to the snooker crossover operator.

Proof. Suppose that the current population $\boldsymbol{x} \sim f(\boldsymbol{x})$ is true, and that x_i and x_j are selected for the snooker crossover. The selection procedure shows that x_i and x_j are independent of one another. Following from Lemma 1, y_i is independent of x_j and has density $f_i(y_i)$. Hence $\boldsymbol{y} \sim f(\boldsymbol{y})$, $f(\boldsymbol{x})$ is invariant with respect to the snooker crossover operator.

3.3 Exchange

The exchange operator is the same as that used in the binary-coded EMC algorithm. Given the current population $\boldsymbol{x}$ and the attached temperature ladder $\boldsymbol{t}$, $(\boldsymbol{x}, \boldsymbol{t}) = (x_1, t_1, \ldots, x_N, t_N)$, we propose to obtain a new population $\boldsymbol{y}$ by making an exchange between x_i and x_j without changing the t's; that is, $(\boldsymbol{y}, \boldsymbol{t}) = (x_1, t_1, \ldots, x_j, t_i, \ldots, x_i, t_j, \ldots, x_N, t_N)$. The new population is accepted with probability $\min(1, r_e)$ according to the Metropolis–Hastings rule,

$$r_e = \frac{f(\boldsymbol{y})}{f(\boldsymbol{x})}\frac{T(\boldsymbol{x}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x})} = \exp\left\{(H(x_i) - H(x_j))\left(\frac{1}{t_i} - \frac{1}{t_j}\right)\right\}. \quad (7)$$

Typically, the exchange is performed only on two individuals with neighboring temperatures; that is, $|i - j| = 1$. Let $p_e(x_i)$ denote the probability that x_i is chosen to exchange with the other individual, and let $p_e(x_j|x_i)$ denote the probability that x_j is chosen to exchange with x_i for a given x_i. Thus we have $T(\boldsymbol{y}|\boldsymbol{x}) = p_e(x_i)p_e(x_j|x_i) + p_e(x_j)p_e(x_i|x_j)$ and $T(\boldsymbol{y}|\boldsymbol{x}) = T(\boldsymbol{x}|\boldsymbol{y})$.

3.4 Algorithm

Based on the operators introduced earlier, the real-coded EMC algorithm is summarized as follows. Given an initial population $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ (initialized at random) and a temperature ladder $\boldsymbol{t} = \{t_1, \ldots, t_N\}$, one iteration comprises two steps:

1. Apply mutation, real crossover or snooker crossover operators to the population with probability p_m, $1 - p_m/2$ and $1 - p_m/2$ (p_m is called a mutation rate).
2. Try to exchange x_i and x_j for N pairs (i, j), with i sampled uniformly on $\{1, \ldots, N\}$ and $j = i \pm 1$ with probability $p_e(x_j|x_i)$, where $p_e(x_{i+1}|x_i) = p_e(x_{i-1}|x_i) = .5$ and $p_e(x_2|x_1) = p_e(x_{N-1}|x_N) = 1$.

In the mutation step, each chromosome of the population is mutated independently. In the crossover step, about 50% of chromosomes are chosen to mate. Note that the crossover operator works in an iterative way; that is, each time, two parental chromosomes are chosen from the updated population by the previous crossover operation.

The algorithm has three user-set parameters: N, $\boldsymbol{t}$, and p_m. For the choice of N and $\boldsymbol{t}$, note that they both are related to the diversity of the population, and that a highly diversed population is always preferred for the system mixing. Indeed, we use two methods to increase the diversity of a population in EMC. One method is to increase the population size N; the other is to steepen the temperature ladder to increase the value of t_1. But these methods need some balance. A small population size may result in a steeper temperature ladder and a low acceptance probability of exchange operations given the temperature range. A large population size may result in x_N being less likely to be updated in per-unit CPU time, and a slow sampler for the target distribution. Neither is an attractive choice. One heuristic guideline is to choose the population size comparable (at least) with the dimension of the problem, choose the highest temperature such that the (anticipated) energy barriers can be easily overcome by a Metropolis move, and choose the temperature ladder to produce a moderate acceptance probability of the exchange operations. For the choice of p_m, note that the mutation operator usually provides a local exploration around a local mode, and the crossover operators usually provide a much more global exploration over the entire sample space and often have a low acceptance probability. To balance the two kinds of operations, we usually set p_m to a small value between .25 and .4, which may not be optimal but usually works well. (For a further discussion of this choice refer to Liang and Wong 2000.)

Each of the foregoing operators, (mutation, crossover, and exchange) leaves the Boltzmann distribution (2) invariant. This implies that the samples of a population are all mutually independent, and that each distribution $f_i(x)$ is also invariant with respect to the operators. Hence the algorithm leads to an effective sampler for a sequence of distributions. Samples can be collected from the corresponding levels for a further inference for the distributions in which we are interested. The algorithm's structure is very flexible. Setting $p_m = 1$, it turns out to be parallel tempering. Setting $p_m = 1$ and $N = 1$, it turns to the single-chain Metropolis–Hastings algorithm.

4. THREE ILLUSTRATIVE EXAMPLES

4.1 A Bimodal Example

We first test the effectiveness of crossover operators on a five-dimensional mixture Gaussian

$$\pi(\boldsymbol{x}) = \frac{1}{3}N_5(\mathbf{0}, I_5) + \frac{2}{3}N_5(\mathbf{5}, I_5),$$

where $\mathbf{0} = (0, \ldots, 0)$ and $\mathbf{5} = (5, \ldots, 5)$, and the distance between the two modes is $5\sqrt{5} = 11.2$. This example is identical to example 6.2 of Liu, Liang, and Wong (2000).

The Metropolis algorithm was first applied to the problem with a proposal distribution unif$[x-2, x+2]^5$. The overall acceptance rate is about .23. In all of the 10^6 iterations, the algorithm could not escape from the mode in which it started.

The real-coded EMC algorithm (hereinafter called simply EMC) was also applied to this problem with a population size of 10 and a constant temperature ladder $t_i \equiv 1$. A cross-test was performed for the snooker crossover and real crossover versus two initial distributions, $N(\mathbf{0}, I_5)$ and $N(\mathbf{2.5}, 25I_5)$, where $\mathbf{2.5} = (2.5, \ldots, 2.5)$. The mutation rate was .25, and e_k was a random draw from unif$[x-2, x+2]^5$. In the snooker crossover step, six parental chromosome pairs were chosen to mate, and the overall acceptance rate was .22. In the real crossover (one-point crossover) step, four parental chromosome pairs were chosen to mate, and the overall acceptance rate was .35. The selection temperature was .1.

Figure 1 shows the sampling paths of the first 2,000 iterations of the four runs. It is seen that if one starts with a locally concentrated population, then the real crossover may fail to

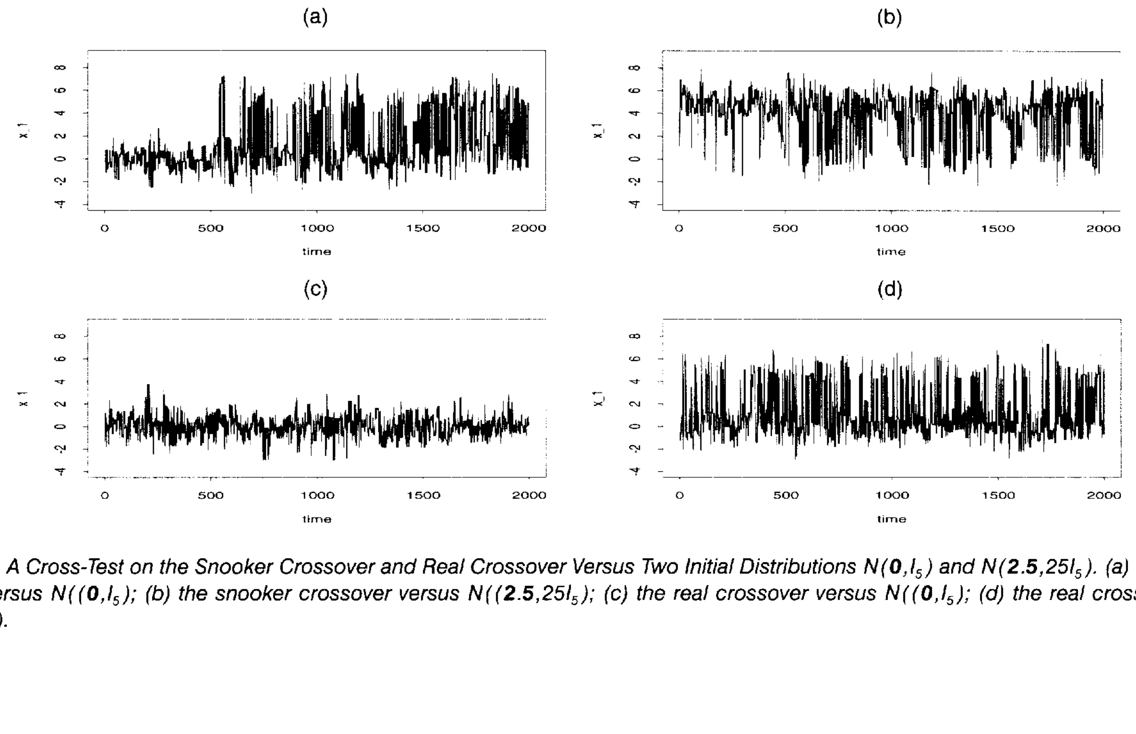

*Figure 1. A Cross-Test on the Snooker Crossover and Real Crossover Versus Two Initial Distributions N(**0**,I₅) and N(**2.5**,25I₅). (a) The snooker crossover versus N((**0**,I₅); (b) the snooker crossover versus N((**2.5**,25I₅); (c) the real crossover versus N((**0**,I₅); (d) the real crossover versus N((**2.5**,25I₅).*

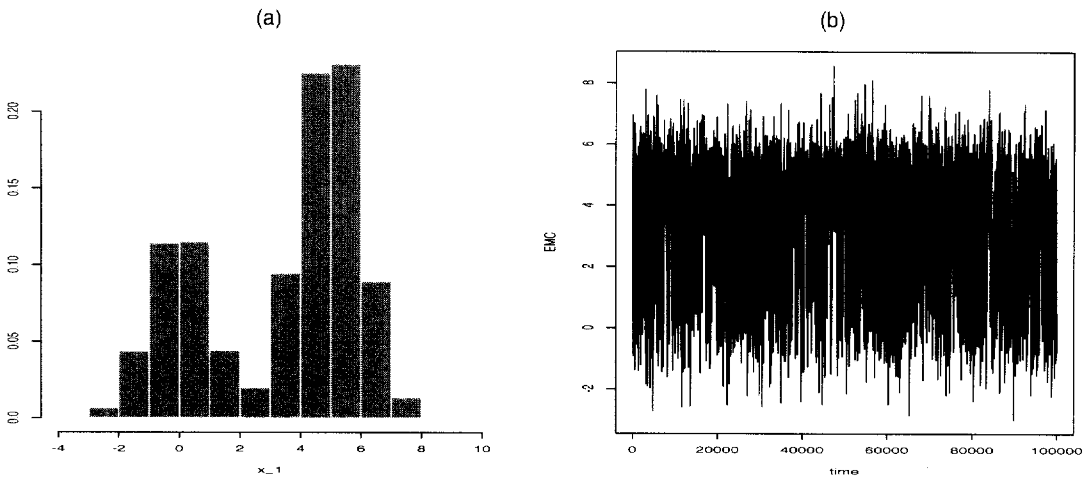

Figure 2. The Mixing Results of the EMC Samples. (a) A marginal histogram of the sample; (b) time series plot of the sample x_N.

escape from the mode in which it was started, but the snooker crossover can still mix well with other modes, although the burn-in time becomes slightly longer. But if one starts with an overdispersed population, then both of the crossover operators can mix well rapidly. Note that the mix of the real crossover in Figure 1(d) is due in part to the exchange operations. With a longer run, the quantities of $\pi(\cdot)$ can be estimated accurately by run (a) or run (b). This experiment clearly shows that the crossover operators provide a much more global exploration over the entire sample space, whereas the mutation operator or the Metropolis move provides a good local exploration around a local mode. The experiment also shows that the snooker crossover is more efficient than the real crossover for this example. Because the real crossover operator introduces a different move, we still include it in the runs of the following examples, although at a small proportion, e.g., .05 or .1. This experiment also suggests that a diversed population is preferred for the mixing of a system. In Section 3 we mentioned that simulating along a temperature is an efficient way to increase the diversity of a population. A run of EMC with a nonconstant temperature ladder is described in the next paragraph.

In this run, the temperature ladder ranged from 5.0 to 1.0 with an equal space; the other parameters had the same setting as in the last run, and only the snooker crossover was used. The overall acceptance probabilities of the mutation, crossover, and exchange operations were .44, .24, and .85. The acceptance probabilities of these operations on f_N were also examined and were .22, .22, and .69. These values imply that EMC has been implemented effectively for this example. With 100,000 iterations, EMC produced 100,000 samples from $\pi(\cdot)$ in about 36 seconds of CPU time on a Sun Ultra2 workstation. The estimates of the mixing proportion, marginal means, variance, and even cdf's of the mixture distribution can be estimated rather accurately (i.e., they differ from the true values in the second decimal place). Figure 2 plots the histogram and the time series of the first coordinate of $\boldsymbol{x}_N$. Compared to CGMC, EMC has created significant savings in computational time for this example. To get the same estimates, CGMC needs about 300 seconds of CPU time on the same machine; this translates to an eight-fold improvement. Note that a relatively large population size and a low selection temperature make EMC perform like CGMC; however, the local optimization procedure is avoided.

4.2 A Multimodal Example

Consider a simulation from a two-dimensional mixture normal distribution,

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}\sum_{i=1}^{20} w_i \exp\left\{-\frac{1}{2\sigma^2}(x-\mu_i)'(x-\mu_i)\right\}, \quad (8)$$

where $\sigma = .1$ and $w_1 = \cdots = w_{20} = .05$. The mean vectors $\mu_1, \mu_2, \ldots, \mu_{20}$ are uniformly drawn from the rectangle $[0, 10] \times [0, 10]$; these are listed in Table 1 and plotted in Figure 3. It is seen that components 2, 4, and 15 are well separated from the others. The distance between component 4 and its nearest-neighboring component is 3.15, and the distance between component 15 and its nearest-neighboring component (except component 2) is 3.84. These distances are 31.5 and 38.4 times of the standard deviation. The mixing of the components across so long a distance puts a great challenge on EMC.

Table 1. The Mean Vectors of the 20 Components of the Mixture Normal Distribution

i	μ_{i1}	μ_{i2}	i	μ_{i1}	μ_{i2}	i	μ_{i1}	μ_{i2}	i	μ_{i1}	μ_{i2}
1	2.18	5.76	6	3.25	3.47	11	5.41	2.65	16	4.93	1.50
2	8.67	9.59	7	1.70	.50	12	2.70	7.88	17	1.83	.09
3	4.24	8.48	8	4.59	5.60	13	4.98	3.70	18	2.26	.31
4	8.41	1.68	9	6.91	5.81	14	1.14	2.39	19	5.54	6.86
5	3.93	8.82	10	6.87	5.40	15	8.33	9.50	20	1.69	8.11

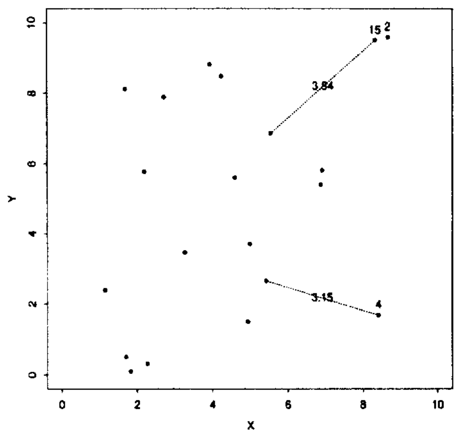

Figure 3. The Center Points of the 20 Components of the Mixture Normal Distribution.

We applied EMC to this problem with the following parameter settings: population size 20, highest temperature 5, lowest temperature 1, intermediate temperatures equally spaced between 5 and 1, and mutation rate .2. The population was initialized by the random vectors sampled uniformly from $U[0, 1] \times U[0, 1]$. The algorithm was run for 100,000 iterations, with a CPU time of 210 seconds (The computation of this example was done on an Ultra Sparc2 workstation). In the mutation step, we chose $e_k \sim N_2(0, .25^2 t_k)$ for $k = 1, \ldots, N$, where t_k is the temperature of the kth level. The overall acceptance rate was .26. The overall acceptance rates of real crossover (one-point crossover) operations and snooker crossover operations were .14 and .16. The overall exchange rate was .95. Figure 4(a) shows the sample paths ($t = 1$) in the first 10,000 iterations; Figure 5(a) shows the whole samples. Figure 4(a) illustrates that EMC has sampled all components in the first 10,000 iterations, although the population was initialized at one corner.

For comparison, we also applied parallel tempering to the example with the same parameter settings and initialization. Within the same computational time frame, parallel tempering produced 73,500 iterations. The local updating step was done by the mutation operator used in EMC, and the acceptance rate was .28. This value suggests that parallel tempering has been implemented effectively. The overall exchange rate was .95. Parallel tempering took longer per iteration, because each chromosome will undergo an updating step. However, only about 50% of chromosomes are chosen to mate in the crossover steps of EMC. Figure 4(b) shows the sample path ($t = 1$) of the first 10,000 iterations, and Figure 5(b) shows the whole samples. Figure 4(b) illustrates that parallel tempering did not sample all components in the first 10,000 iterations, and that most samples got stuck at the starting corner. Figure 5(b) shows that parallel tempering could not sample correctly from the mixture distribution even with 73,500 iterations; the components 2, 4, and 15 were never sampled.

For further comparison, we ran both algorithms 20 times independently. Each run of EMC consists of 10^6 iterations, each run of parallel tempering consists of 7.35×10^5 iterations, and they have about the same CPU time. The mean and variance of the mixture distribution were estimated; the results are summarized in Table 2 (evolutionary A and parallel tempering). It is clear that EMC has estimated the mean and variance rather accurately, but that parallel tempering fails completely.

In another experiment, we examined the effect of the snooker crossover. The evolutionary algorithm with only the real crossover operator was run 20 times. Each run comprised of 10^6 iterations, and the CPU time was about the same as the last experiment. The estimation results were shown in Table 2 (evolutionary B). The comparison shows that the snooker crossover is superior to the real crossover and has a faster convergence rate for simulation.

4.3 Galaxy Data Example

The dataset for this example comprises velocities of 82 galaxies from six well-separated conic sections of the corona borealis region. It was first presented by Postman, Huchra, and Geller (1986), and subsequently was analyzed by many others, including Roeder (1990), Carlin and Chib (1993), Chib (1995), Phillips and Smith (1995), and Neal (1999). Our objective is to find the best-fitting Gaussian finite-mixture model. Following Chib (1995), we use the following setup for this example. Let $y_1, \ldots, y_n$ denote the iid samples from a mixture model of d components. The density function of the jth component

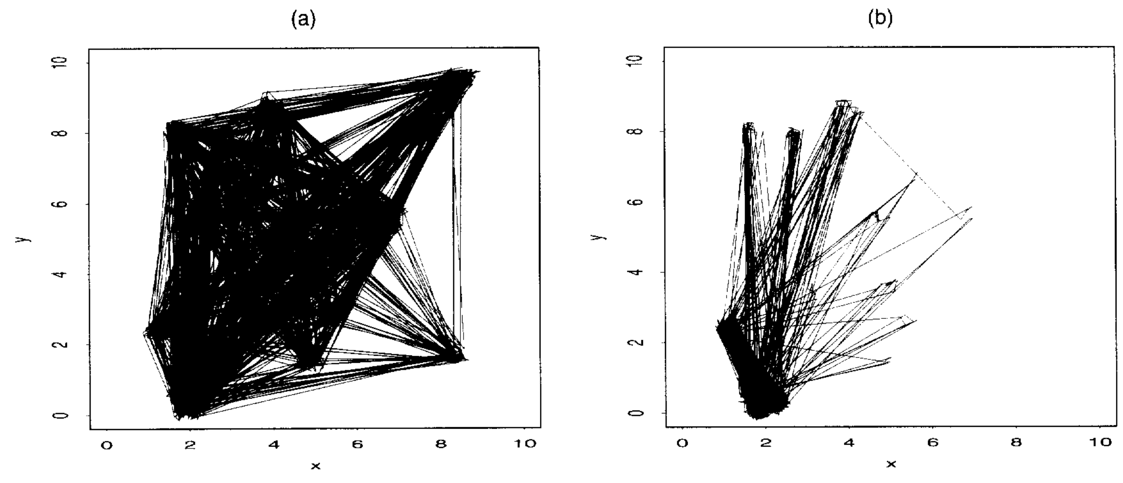

Figure 4. Sample Path of the First 10,000 Iterations at Temperature $t = 1$. (a) EMC; (b) parallel tempering.

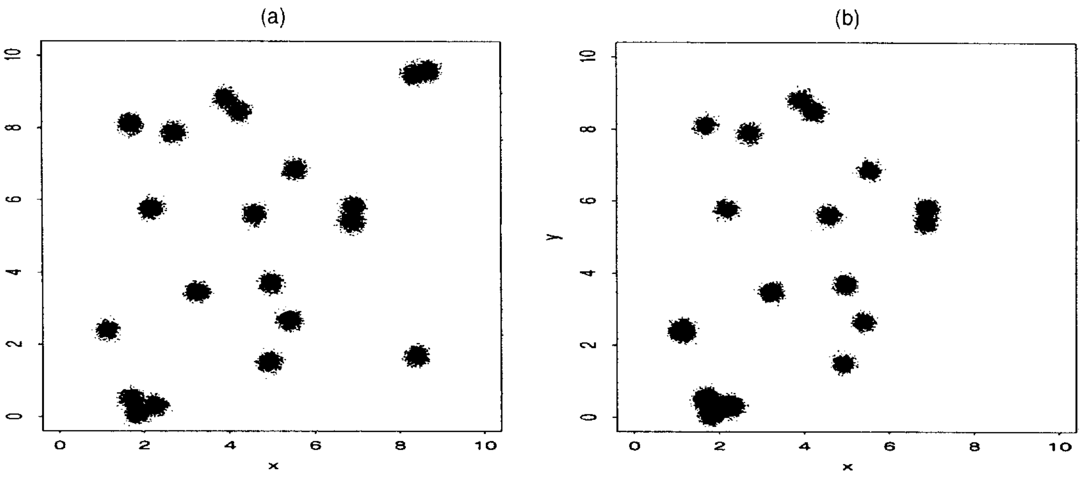

Figure 5. The Plot of Whole Samples. (a) EMC; (b) parallel tempering.

is given by $\phi(y|\mu_j, \sigma_j^2)$, where (μ_j, σ_j^2) is the component-specific mean and variance and the mixture proportion is denoted by p_j $(\sum_{j=1}^{d} p_j = 1)$. The likelihood function is then

$$L(y|\theta^{(d)}, M_d) = \prod_{i=1}^{n}\sum_{j=1}^{d} p_j\phi(y_i|\mu_j, \sigma_j^2),$$

where $\theta^{(d)} = (p_1, \ldots, p_{d-1}, \mu_1, \sigma_1^2, \ldots, \mu_d, \sigma_d^2)$ is the set of parameters of model M_d. Assume that all components of θ are mutually independent and have the prior distributions

$$\mu_j \sim N(\mu_0, A^{-1}),$$

$$\sigma_j^2 \sim \mathrm{IG}\left(\frac{\nu_0}{2}, \frac{\delta_0}{2}\right),$$

and

$$\boldsymbol{q} \sim \text{Dirichlet}\,(\alpha_1, \ldots, \alpha_d),$$

where $\mu_0 = 20$, $A^{-1} = 100$, $\nu_0 = 6$, $\delta_0 = 40$, $\boldsymbol{q} = (p_1, \ldots, p_{d-1}, 1 - \sum_{j=1}^{d-1} p_j)$, and $\alpha_1 = \cdots = \alpha_d = 1$.

The marginal likelihood under model M_d is

$$m(y|M_d) = \int L(y|\theta^{(d)}, M_d)\pi(\theta^{(d)}|M_d)d\theta^{(d)},$$

where $\pi(\theta^{(d)}|M_d)$ denotes the prior density of $\theta^{(d)}$. Typically, the foregoing integration has no analytical form and must be evaluated by numerical or MC methods. A recent review of the methods was given by Chen et al. (2000). In this article we show that the marginal likelihood can be evaluated conveniently and efficiently from the outputs of EMC using bridge sampling (Meng and Wong 1996).

Bridge sampling works as follows. Suppose that we have two distributions f_i $(i = 1, 2)$, and that each is known up to a multiplicative constant c_i; that is, $f_i(w) = g_i(w)/c_i$.

Let $w_{i1}, \ldots, w_{in_i}$ denote the samples drawn from $f_i(w)$, $i = 1, 2$. The ratio $r = c_1/c_2$ can be estimated iteratively through the following formula:

$$\hat{r}^{(t+1)} = \frac{\frac{1}{n_2}\sum_{j=1}^{n_2} \frac{l_{2j}}{s_1 l_{2j} + s_2\hat{r}^{(t)}}}{\frac{1}{n_1}\sum_{j=1}^{n_1} \frac{1}{s_1 l_{1j} + s_2\hat{r}^{(t)}}},$$

where $s_i = n_i/(n_1 + n_2)$ and $l_{ij} = g_1(w_{ij})/g_2(w_{ij})$ $(j = 1, \ldots, n_i, i = 1, 2)$, which need to be computed only once at the beginning of iterations, and $\hat{r}^{(t+1)}$ denotes the estimated value of r in the $(t+1)$st iteration. The initial guess $\hat{r}^{(0)}$ can be any value larger than 0. This estimate is consistent (Meng and Wong 1996). Let $g_1 = L(y|\theta^{(d)}, M_d)\pi(\theta^{(d)}|M_d)$ and $g_2 = \pi(\theta^{(d)}|M_d)$; the resulting multiplicative constants ratio turns out to be the marginal likelihood $\pi(y|M_d)$.

Motivated by the foregoing choice of g_1 and g_2, we have the following distribution ladder for EMC:

$$f_i(x) \propto [L(y|\theta^{(d)}, M_d)]^{u_i}\pi(\theta^{(d)}|M_d),$$

for $i = 0, 1, \ldots, N$, where u_i denotes the inverse of temperature and $0 = u_0 < u_1 < \cdots < u_N = 1$. Note that $f_0(x)$ corresponds to $\pi(\theta^{(d)}|M_d)$, which can be sampled from directly. The other distributions $f_i(x)$ $(i = 1, \ldots, N)$ can

Table 2. Parameter Estimates Based on the Samples Over 20 Independent Runs

		Evolutionary A		Evolutionary B		Parallel tempering	
Parameter	True value	Estimate	SD	Estimate	SD	Estimate	SD
μ_1	4.478	4.481	.0043	4.444	.0259	3.781	.0316
μ_2	4.905	4.909	.0076	4.862	.0230	4.337	.0435
Σ_{11}	5.552	5.549	.0062	5.544	.0507	3.656	.1114
Σ_{22}	9.861	9.841	.0097	9.775	.0481	8.546	.0485
Σ_{12}	2.605	2.591	.0105	2.580	.0434	1.294	.0839

NOTE: Here μ_1 and μ_2 denote the first and second component of the mean vector of distribution (8) Σ_{11}, Σ_{22}, and Σ_{12} denote three components of the covariance matrix of distribution (8); and SD denotes the standard deviation of the corresponding estimate. Evolutionary A: with both the real crossover and the snooker crossover; evolutionary B: with only the real crossover.

be sampled from using EMC, and the marginal likelihood $m(y|M_d)$ then can be estimated by

$$\hat{r} = \prod_{i=1}^{N} \hat{r}_{i-1,i},$$

where $\hat{r}_{i-1,i}$ denotes the estimated multiplicative constant ratio between f_i and f_{i-1}. The posterior distribution $(f_N(x))$ of the mixture model is invariably a multimodal function; for example, the multimodality can be caused simply by a random permutation of model components, and the resulting modes are often isolated from each other. Sampling from such a multimodal distribution poses a great challenge for EMC.

In our simulation, $N = 20$, $u_1 = .05$, $u_N = 1$, and the other values of u_i $(i = 2, \ldots, N-1)$ are equally spaced between u_1 and u_N. The mutation rate is .4. Models with two to five components are considered. For each model, we ran EMC 20 times independently, with each run comprising 25,000 iterations. The first 5,000 iterations were discarded for the burn-in process. Figure 6 shows the pairwise scatterplot of the component means sampled by EMC for a three-component model. Clearly, EMC has sampled from all possible modes.

Table 3 lists the estimated log-marginal likelihood values for the models with $d = 2$ to 5. For comparison, it also lists the computational results of Chib (1995) and Neal (1999). Chib estimated the log-marginal likelihood from the Gibbs output. Neal estimated the log-marginal likelihood by simple Monte Carlo integration with 10^8 points drawn from the prior distribution. The results show that a model with three-component models fit well to the data, and the four- and five-component models overfit to the data. This result is consistent with the findings of Chib (1995).

In theory, the marginal likelihood values should be estimated based on the samples from all modes of the posterior distribution. But we know that some modes are caused simply by random permutations of model components, and if we impose a relabeling constraint (e.g., $\mu_1 < \mu_2 \cdots < \mu_d$), on the parameter space, then the number of posterior modes will be reduced by a factor of $d!$. An interesting question is whether the marginal likelihood value depends on the factor $d!$—in other words, if the marginal likelihood can be computed based on the samples from only a space restricted by the relabeling constraint. For a test, we impose the relabeling constraint on a three-component model. The estimation result shows that the relabeling constraint does not change the marginal likelihood value. This can also be determined from our marginal likelihood estimation method, which depends only on the likelihood values and is invariant with respect to the relabeling constraint.

Due to the complexity of the likelihood evaluation of this example, the CPU time of each iteration is dominated by that used for the likelihood evaluations. Thus the total CPU time of the simulation is approximately proportional to that of likelihood evaluations. With the above parameter setting, on average the likelihood will be evaluated 12.5 times in one iteration of EMC (five pairs of chromosomes are chosen to mate in the crossover step), so each estimator of EMC in Table 3 is based on 6.25×10^6 likelihood evaluations. Compared to the simple Monte Carlo integration where 10^8 evaluations are used, significant computational savings has been created by EMC. This point is made more clear as the value of d increases. Note that here the estimates of EMC may not be directly comparable with the estimate of Chib (1995), who imposed a relabeling constraint on the parameter space, sampled from only one mode, and used a smaller sample size.

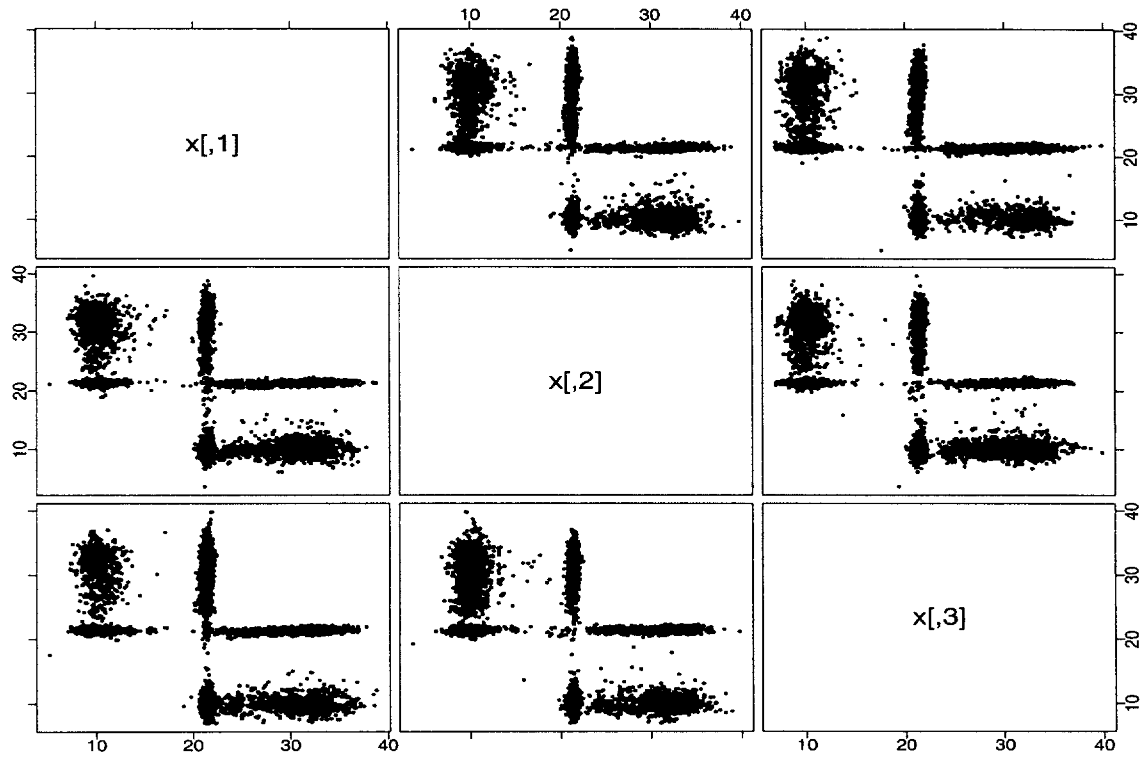

Figure 6. Pairwise Scatterplot of the Component Means of the Samples From Evolutionary Monte Carlo in One Run With 20,000 Iterations.

Table 3. Summary of the Results for Galaxy Data

Model fitted	*Chibs' estimate*	*Neals' estimate*	*EMC*
Two components, equal variances	−240.464 (.006)	−239.764 (.005)	−239.744 (.015)
Three components, equal variances	−228.620 (.008)	−226.803 (.040)	−226.828 (.061)
Three components, unequal variances	−224.138[a] (.086)	−226.791 (.089)	−226.780 (.058)
Three[b] components, unequal variances			−226.768 (.057)
Four components, unequal variances			−226.629 (.061)
Five components, unequal variances			−226.394 (.062)

NOTE: The log-marginal likelihood values are estimated based on 20 independent runs.
[a]According to an anonymous JASA referee, the figure of −224.138 is a "typo," with the correct figure being −228.608 (Neal 1999).
[b]A relabeling constraint ($\mu_1 \le \mu_2 \le \mu_3$) is put on the model such that the component means are in an increasing order in the simulation.

5. TWO BAYESIAN NEURAL NETWORK EXAMPLES

Suppose that we have data pairs $\mathbf{D} = \{(y_1, \boldsymbol{x}_1), (y_2, \boldsymbol{x}_2), \ldots, (y_n, \boldsymbol{x}_n)\}$, which are generated from the relationship

$$y_t = f(\boldsymbol{x}_t) + \epsilon_t, \tag{9}$$

where $y_t \in R^1, \boldsymbol{x}_t \in R^p$, and $\epsilon_t \sim N(0, \sigma^2)$ for $t = 1, \ldots, n$. The function $f(\cdot)$ is unknown and may be highly nonlinear, and the approximation to which has been one of central topics in statistics. In this article we propose to approximate it using a one-hidden layer feed-forward neural network model,

$$\hat{f}(\boldsymbol{x}_t) = \sum_{j=1}^{M} \beta_j \psi(\boldsymbol{x}_t' \gamma_j + \eta_j), \tag{10}$$

where $M \in N$ denotes the number of hidden units, $\beta_j \in R$ and $\gamma_j \in R^p$ denote the connection weights from the hidden unit j to the output unit and the connection weights from the input units to the hidden unit j, and η_j denotes the bias term of the hidden unit j and can be viewed as a connection weight from an additional input unit with a constant input, say $x_0 = 1$. In the following, we suppress the bias term by treating it as a component of γ_j. The activation function $\psi(\cdot)$ is a tanh function,

$$\psi(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}. \tag{11}$$

The interest in the neural network model comes from its universal approximation property (Cybenko 1989; Hornik, Stinchcombe, and White 1989; White 1992); that is, a neural network with a sufficient number of hidden units and properly adjusted connection weights can approximate most functions (including any continuous function with a bounded support) arbitrarily well. Over the last several decades, many algorithms have been proposed to train neural networks, including conjugate gradient, back propagation (Rumelhart, Hinton, and Williams 1986), and their variants.

The Bayesian neural network model was first introduced by Buntine and Weigend (1991) and MacKay (1992). Later, it was analyzed by Neal (1993, 1996) with hybrid Monte Carlo. Recently, Müller and Insua (1998) applied the Gibbs sampler (Geman and Geman 1984) and reversible jump MCMC (Green 1995) to it. In this article we view the model as a nonlinear regression of response y on the covariates $\boldsymbol{x}$,

$$y_t = \sum_{j=1}^{M} \beta_j \psi(\boldsymbol{x}_t' \gamma_j) + \epsilon_t, \tag{12}$$

where $\epsilon_t \sim N(0, \sigma^2)$ for $t = 1, \ldots, n$. The prior distributions are assumed to be $\beta_j \sim N(0, \sigma_\beta^2)$, $\gamma_j \sim N(0, \sigma_\gamma^2 I)$ for $j = 1, \ldots, M$, $\sigma^{-2} \sim \text{gamma}(\nu, \delta)$. The log-posterior (up to an additive constant) of the model is

$$\begin{aligned}\log \pi(\beta, \gamma, \sigma^{-2}|D) \propto& -\left(\frac{n}{2} + \nu - 1\right)\log(\sigma^2) \\ &- \frac{1}{2\sigma^2}\left\{2\delta + \sum_{t=1}^{n}\left[y_t - \sum_{j=1}^{M}\beta_j\psi(\boldsymbol{x}_t'\gamma_j)\right]^2\right\} \\ &- \sum_{j=1}^{M}\frac{\beta_j^2}{2\sigma_\beta^2} - \sum_{j=1}^{M}\sum_{i=0}^{p}\frac{\gamma_{ij}^2}{2\sigma_\gamma^2}.\end{aligned} \tag{13}$$

The posterior distribution is invariably an extremely complex function. The complexity is caused mainly by two factors, the nonlinearity and the multimodality. For example, the posterior distribution is invariant with respect to arbitrary relabeling of the hidden units. Because $\psi(-x) = -\psi(x)$, it is also invariant with respect to the simultaneous changes of the signs of β_j and γ_j for some value of j. One way of avoiding the multimodality is to impose a constraint on the parameter space; for example, $0 < \gamma_{11} < \gamma_{21} < \cdots < \gamma_{M1}$ (Müller and Insua 1998). In this article we impose no constraint on the parameter space, so the example is meant to illustrate how the evolutionary algorithm performs in presence of multimodality and nonlinearity.

Under the Bayesian framework, the point prediction can be obtained by integrating out the nuisance parameters,

$$\begin{aligned}E(y_{t+1}|\boldsymbol{x}_{t+1}, D) = \int\int\int h(\boldsymbol{x}_{t+1}, \beta, \gamma, \sigma^{-2}) \\ \times \pi(\beta, \gamma, \sigma^{-2}|D)d\beta d\gamma d\sigma^{-2},\end{aligned} \tag{14}$$

where $h(\boldsymbol{x}_{t+1}, \beta, \gamma, \sigma^{-2})$ denotes a point prediction at $\boldsymbol{x}_{t+1}$ based on a single sample of $(\beta, \gamma, \sigma^{-2})$. Note when $\boldsymbol{x}_{t+1}$ is observable, we have

$$h(\boldsymbol{x}_{t+1}, \beta, \gamma, \sigma^{-2}) = \sum_{j=1}^{M} \beta_j \psi(\boldsymbol{x}_{t+1}' \gamma_j). \tag{15}$$

The resulting prediction is unbiased.

Table 4. The Connection Weights Corresponding to the Eight Modes of the Posterior Distribution of the Simulated Example

No.	γ_{10}	γ_{11}	γ_{20}	γ_{21}	β_1	β_2
1	2	−1	1	1.5	20	10
2	−2	1	1	1.5	−20	10
3	−2	1	−1	−1.5	−20	−10
4	2	−1	−1	−1.5	20	−10
5	1	1.5	2	−1	10	20
6	−1	−1.5	2	−1	−10	20
7	−1	−1.5	−2	1	−10	−20
8	1	1.5	−2	1	10	−20

5.1 A Simulated Example

We simulated $y_1, \ldots, y_{100}$ from (12) with $M = 2$, $\gamma_1 = (\gamma_{10}, \gamma_{11}) = (2, -1)$, $\gamma_2 = (\gamma_{20}, \gamma_{21}) = (1, 1.5), \beta = (20, 10), \sigma = .1$, and the input pattern $\boldsymbol{x}_t = (1, z_t)$, where $z_t = t * .1$ for $t = 1, 2, \ldots, 100$.

This example is identical to example 2 of Müller and Insua (1998) except that there a sigmoidal function was used as the activation function. The posterior distribution of the example has at least eight modes, because of the arbitrary permutation of the hidden units and the simultaneous changes of the signs of γ and β. The connection weights corresponding to the eight modes are listed in Table 4.

In EMC, a neural network model with two input units, two hidden units, and one output unit (a 2-2-1 structure) was simulated. The prior parameters were set as $\sigma_\beta = 20, \sigma_\gamma = 5, \nu = .01$, and $\delta = .01$. The population size was 20. The highest temperature was 20, the lowest temperature was .1, and the intermediate temperatures were equally spaced between 20 and .1. The selection temperature was .01. The mutation rate was .25. The lowest temperature is sufficiently low that the samples on that level will cluster in the small neighborhoods of each mode, and it is meant to test whether the global optima can be located by EMC as the temperature tends to 0. For comparison, we also applied parallel tempering to the example with the same neural network structure and the same parameter settings.

We ran the two algorithms five times independently on an Ultra Sparc2 workstation. Each run of EMC comprised 20,000 iterations, and the CPU time was 152 seconds. The overall acceptance rates of the mutation, real crossover (one-point crossover), snooker crossover, and exchange operations were .07, .09, .04 and .72. With the same CPU time, parallel tempering was iterated 14,000 times. The overall acceptance rates of the local Metropolis moves and exchange were .13 and .66. The low acceptance rate of the local Metropolis moves results from the fact that the system always got stuck in local minima in the last several thousands of iterations. The acceptance rate was .24 in the first 5,000 iterations.

Figure 7(a) shows the negative of the log-posterior values of the samples at $t = .1$ versus the computational time. Note that the two algorithms have been adjusted to have the same time scale (i.e., we plot a point every 20 iterations for EMC and every 14 iterations for parallel tempering). Figure 7(b) plots the mean squared error (MSE) values corresponding to the runs in Figure 7(a). The similarity of these two plots illustrates the equivalence of maximizing a posterior distribution and minimizing the MSE for training a neural network. The latter is often used as an objective when training non-Bayesian neural networks. Later, we ran the two algorithms 100 times independently; 97 runs of EMC converged to some value below -160, but no run of parallel tempering converged to a value below -160, and all of them got stuck at some local minima.

Figure 8 shows the maximum a posteriori (MAP) estimate of the regression line obtained in one run. Figures 9(a) and 9(b) plot the histograms γ and β. The samples come from 750 independent runs of EMC. It is clear that all eight possible modes have been sampled by EMC, and the relative weights among them are also estimated roughly correctly (these modes are equally weighted). These results demonstrate the superiority of EMC over parallel tempering in sampling from a distribution with multimodality and nonlinearity. They also show that EMC may possibly work as an optimization algorithm as t_N tends to 0.

5.2 A Real Example

The dataset for this example is the gas furnace data from Box and Jenkins (1970). The time series consists of 296 pairs

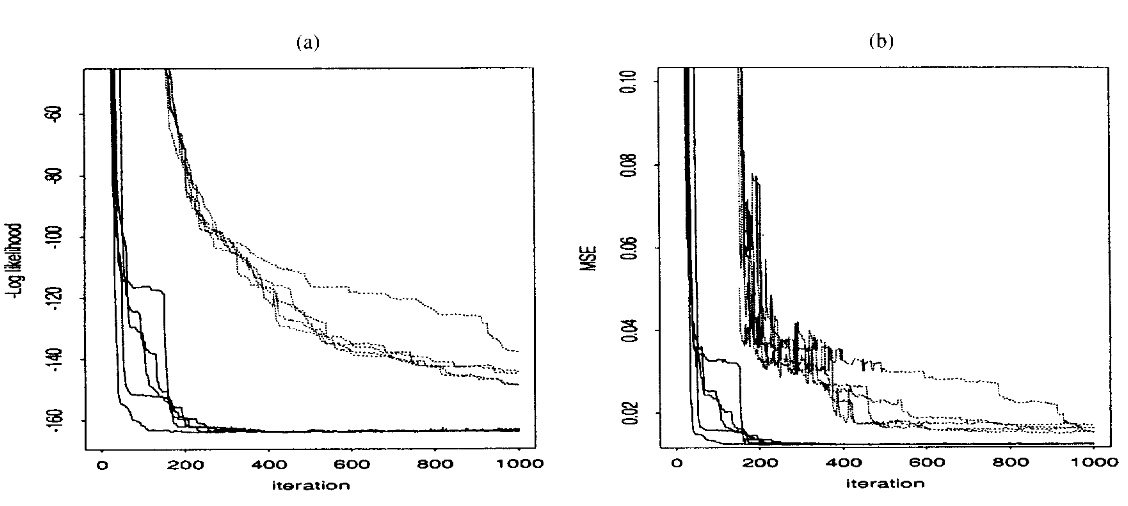

Figure 7. Comparison of EMC (———) and Parallel Tempering (···). (a) The negative of log-posterior values of the samples versus the computational time; (b) the MSE values of the samples versus the computational time.

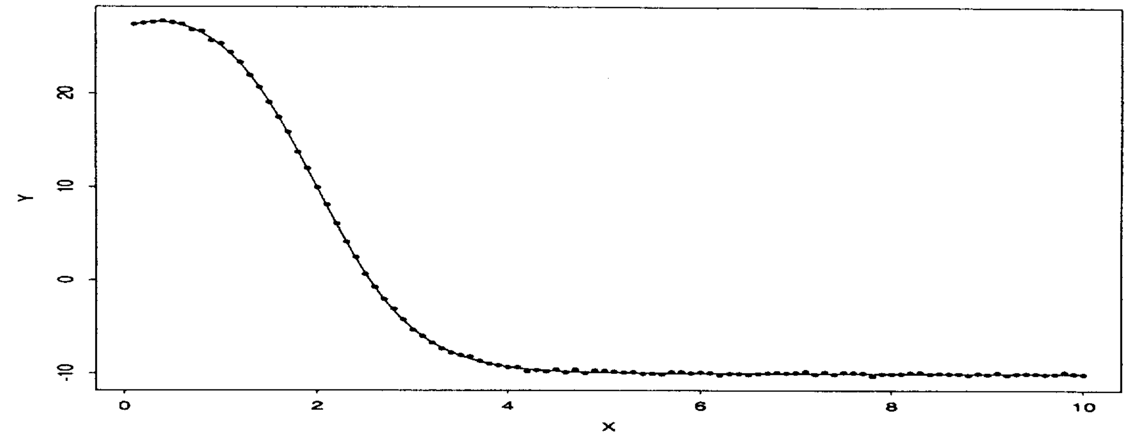

Figure 8. The Original Data (···) and the MAP Estimate (———) of the Nonlinear Regression Line.

of input–output observations. The input x_t is the gas flow rate into a furnace, and the output y_t is the CO_2 concentration from the furnace. The sampling rate is 9 seconds. Because our focus was to illustrate the usefulness of EMC in training and forecasting of Bayesian neural networks, we used only the output y_t and treated it as a univariate time series.

In this problem, a neural network with a more general structure was used,

$$y_t = \boldsymbol{x}_t'\lambda + \sum_{j=1}^{M} \beta_j \psi(\boldsymbol{x}_t'\gamma_j) + \epsilon_t, \tag{16}$$

where $\boldsymbol{x}_t = (1, y_{t-1}, \ldots, y_{t-p})$, p is the number of input units, and λ denotes a vector of shortcut connections; that is, the connections from the input units to the output unit. An independent normal prior is imposed on each λ_k; that is, $\lambda_k \sim N(0, \sigma_\lambda^2)$ for $k = 0, \ldots, p$. The other parameters are subject to the same prior setting as in the preceding example. The log-posterior (up to an additive constant) of the model is

$$\begin{aligned}
&\log \pi(\lambda, \beta, \gamma, \sigma^2 | D) \\
&\quad \propto -\left(\frac{n}{2} + \nu - 1\right) \log(\sigma^2) \\
&\quad -\frac{1}{2\sigma^2}\left\{2\delta + \sum_{t=1}^{n}\left[y_t - \boldsymbol{x}_t'\lambda - \sum_{j=1}^{M}\beta_j\psi(\boldsymbol{x}_t'\gamma_j)\right]^2\right\} \\
&\quad -\sum_{i=0}^{p}\frac{\lambda_i^2}{2\sigma_\lambda^2} - \sum_{j=1}^{M}\frac{\beta_j^2}{2\sigma_\beta^2} - \sum_{j=1}^{M}\sum_{i=0}^{p}\frac{\gamma_{ij}^2}{2\sigma_\gamma^2}.
\end{aligned} \tag{17}$$

Following Box and Jenkins (1970), we used the first 206 observations as the training data and used the remaining observations as for test. A neural network with a 4–5–1 structure and shortcut connections was simulated with EMC. This network model has 35 tunable connection parameters plus σ^2, for a total of 36 unknown parameters. Thus this example illustrates EMC's ability to sample from a distribution defined on a high-dimensional space.

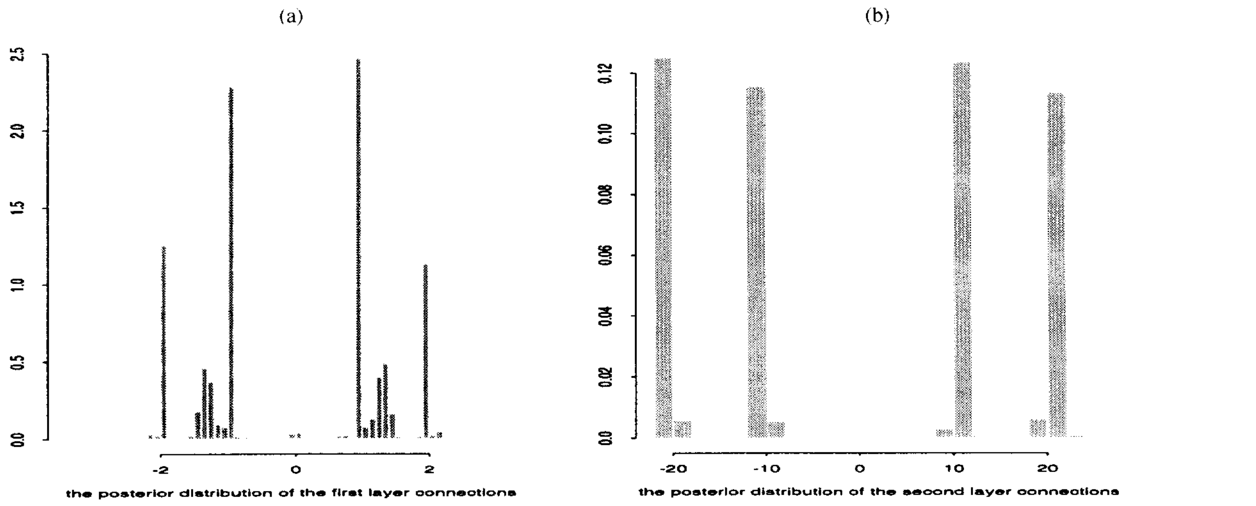

Figure 9. Histograms of the Posterior Distribution of the Network Connections. (a) The first layer connections γ; (b) the second layer connections β.

The prior parameters were set as $\sigma_\lambda = \sigma_\beta = \sigma_\gamma = 10$ and $\nu = \delta = .05$. The population size was 40. The highest temperature was 15, the lowest temperature was 1, and the intermediate temperatures were equally spaced between 15 and 1. The selection temperature was .1, and the mutation rate was .25.

First, the evolutionary algorithm was run five times independently. Each run comprised 2.5×10^5 iterations done on an Alpha500 workstation, and the CPU time was 60 minutes. The overall acceptance rate of mutation, real crossover (one-point crossover), snooker crossover, and exchange operations were .11, .03, .51, and .79.

For comparison, parallel tempering, conjugate gradient, and the Box–Jenkins approach were also applied to this example. The training errors associated with these methods are shown in Table 5. In parallel tempering, we used the same neural network structure and the same parameter setting as in EMC. We ran parallel tempering five times independently, with each run comprising 1.5×10^5 iterations and a CPU time of 73 minutes, 20% longer than that of EMC.

The conjugate gradient method trains a neural network by minimizing the total fitting error $\sum_{t=1}^{n}(y_t - \hat{y}_t)^2$, where n is the number of training cases. This method is similar to back propagation (Rumelhart et al. 1986), as both are gradient based. The efficiency of the conjugate gradient method in training neural networks has been widely acknowledged (Mehrotra, Mohan, and Ranka, 1996). In this example, we ran the method until the convergence occurred. Considering the dependence of the performance of the method on the starting values, we ran it 100 times independently with different initial connection weights, with each sampled uniformly on the hypercube $[-.1, .1]^{35}$. The conjugate gradient method converged very fast in the example. On average, the number of iterations to convergence was 164.4. The total CPU time for the 100 runs was 163 minutes, 2.7 times longer than that of EMC.

Following Box and Jenkins (1970), we fitted an ARMA(4, 2) model using the Box–Jenkins approach. This was done via the standard procedure "arima.mle" in S-PLUS.

One important goal of time series analysis is to forecast future values. For the one-step-ahead prediction, the covariate $\boldsymbol{x}_{t+1}$ is observable, and y_{t+1} can be predicted unbiasedly using (14). For the multiple-step-ahead prediction, we have the following recursive procedure (Ding, Canu, and Denoeux 1996; Hill, O'Connor, and Remus 1996). Given a sequence of samples of neural networks $(\beta_1, \gamma_1), \ldots, (\beta_m, \gamma_m)$, the l-step-ahead prediction involves two steps:

1. For each sample (β_i, γ_i), forecast y_{t+1} using (15), then use the forecast value as a new input to forecast y_{t+2}, and so on, until y_{t+l} is predicted.
2. Average the forecast values of y_{t+l} over the neural network samples to get the final forecast value of y_{t+l}.

Although the prediction is biased, the procedure is very simple and intuitive, and performs well in most cases. The prediction ability (generalizability) of neural networks can be simply measured by the mean squared prediction error (MSPE),

$$\mathrm{MSPE}(l) = \sum_{T=t}^{n-l}[y_{T+l} - \hat{y}_{T+l}(T)]^2/(n-l-t+1), \qquad (18)$$

Table 5. Comparison of the MSPE's of the Four Methods.

		MSPE		
Run	*Training error*	*One-step*	*Two-step*	*Three-step*
(a) Box-Jenkins approach				
	.09594	.1569	.8873	2.4638
(b) Conjugate gradient				
1	.09471	.1558	.8571	2.3576
2	.09472	.1569	.8670	2.3947
3	.09472	.1542	.8429	2.3068
4	.09472	.1558	.8587	2.3677
5	.09473	.1567	.8675	2.4017
Avg.	.09472	.1559	.8586	2.3657
(c) Simulated tempering				
1	.09463	.1566	.8644	2.3868
2	.09457	.1569	.8650	2.3858
3	.09466	.1566	.8633	2.3819
4	.09436	.1569	.8646	2.3778
5	.09432	.1566	.8605	2.3477
Avg.	.09451	.1567	.8636	2.3760
(d) Evolutionary sampling				
1	.08692	.1612	.8060	2.0747
2	.08408	.1504	.7926	2.1058
3	.08711	.1636	.7761	1.9103
4	.08609	.1523	.7720	1.9921
5	.08694	.1569	.7902	2.0171
Avg.	.08623	.1569	.7874	2.0200

NOTE: The MSPEs are calculated with $T = 207$ and $n = 296$. (a) The Box–Jenkins approach; (b) the MSPEs of the conjugate gradient method in five (out of 100) runs with the smallest training errors; (c) the MSPEs of parallel tempering in five runs; (d) the MSPEs of the evolutionary algorithm in five runs.

where l denotes the l-step-ahead prediction, and $\hat{y}_{T+l}(T)$ denotes the point prediction of y_{T+l} made at time T.

Table 5 gives MSPE(l)s ($l = 1, 2, 3$) of the four methods. For all methods, the MSPE(1)s are comparable. However, MSPE(2) and MSPE(3) of the Box–Jenkins approach are larger than those of the other three neural network–based methods. It is not surprising that as the forecast horizon extends, the Box–Jenkins model performs less well, because this model is best suited for short-term forecasting. This is also consistent with the finding of Hill et al. (1996) and Kang (1991) that neural network models generally perform better in the latter periods of the forecast horizon. Among the three neural network–based methods, EMC clearly outperforms the other two methods, conjugate gradient and parallel tempering. The small training and prediction errors of EMC implies that an efficient sampler is an essential tool for improving the generalizability of Bayesian neural networks.

6. CONCLUSIONS

This article extends the EMC method to sample from a target distribution with real-valued parameters, using the snooker operator to implement the crossover operations. Simulation studies confirm the method's effectiveness. The effectiveness is due to two factors: the method has incorporated the learning ability of the genetic algorithm by evolving with crossover operators, and it has incorporated the fast mixing ability of parallel tempering (simulated tempering) by simulating along a temperature ladder. We posit the following explanation for EMC's learning capability.

In general, a "learning" capability means that one is able to modify a behavioral tendency by experience. For a MCMC sampler, this means that the sampler can be guided by the samples obtained in previous steps. Clearly, the single-chain-based MCMC algorithms (e.g., the Metropolis algorithm and the Gibbs sampler) have no such an ability because of their Markovian properties. The use of population, which works as the state space of the Markov chain and also as a dynamic memory of the simulation, provides an opportunity for the MCMC sampler to learn from its historical samples.

In parallel tempering, all samples are simulated in parallel, no crossover operation is used, and no "learning" occurs between samples. In adaptive direction sampling, all samples of the population are iid. At each step, one sample (the current sample) is randomly selected to update along a direction pointing to another sample (the anchor sample) of the current population. Usually the anchor sample is chosen to be iid with the current sample, and the "learning" is not managed in an efficient way. CGMC does have a learning capability, but it learns from a local mode found by the local optimization procedure instead of the historical samples. We realize that a local optimization procedure does increase the diversity of the population and improve the mixing rate of a system. We suggest possibly incorporating a CGMC sampler into EMC as a crossover operator when the local optimization procedure is available for the problem. Because of the computational cost of local optimization, it can be applied at a low frequency (e.g., .05 or .1).

The setting of EMC facilities its learning from the historical samples. The simulation at high temperatures can help the system explore the entire sample space. The exchange operation (swapping of temperature) can be viewed as a selection mechanism. A "bad" sample (with a low fitness value) will, through the exchange operations, be forced to climb up the temperature ladder. At high temperatures, random mutations are easily accepted, and thus the sample will be easily eliminated from the population. In contrast, a "good" sample (with a high fitness value) will be forced to climb down the temperature ladder. At low temperatures, random mutations are accepted with difficulty, and the sample will be stored there for a relatively long time. According to the roulette wheel selection procedure used by EMC, the "good" sample will have a high probability of being selected as a parental sample for mating to produce offspring, which will have a high probability of resembling the parental sample. In this sense, we say that EMC has learned from its historical samples. The temperature ladder together with the exchange operator conveniently provides a selection mechanism for the dynamic memory of EMC.

[Received October 1999. Revised July 2000.]

REFERENCES

Berg, B. A., and Neuhaus, T. (1991), "Multicanonical Algorithms for First Order Phase Transitions," *Physics Letters*, Ser. B, 267, 249–253.

Box, G. E. P., and Jenkins, G. M. (1970), *Time Series Analysis, Forecast and Control*, San Francisco: Holden-Day.

Buntine, W. L., and Weigend, A. S. (1991), "Bayesian Back-Propagation," *Complex Systems 5*, 603–643.

Carlin, B., and Chib, S. (1993), "Bayesian Model Choice via Markov Chain Monte Carlo," *Journal of the Royal Statistical Society*, Ser. B, 57, 473–484.

Chen, M. H., Shao, Q. M., and Ibrahim, J. G. (2000), *Monte Carlo Methods in Bayesian Computation*, New York: Springer-Verlag.

Chib, S. (1995), "Marginal Likelihood From the Gibbs Output," *Journal of American Statistical Association*, 90, 1313–1321.

Cybenko, G. (1989), "Approximations by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals and Systems*, 2, 303–314.

Ding, X., Canu, S., and Denoeux, T. (1996), "Neural Network Based Models for Forecasting," in *Neural Networks and Their Applications*, ed. J. G. Taylor, New York: Wiley, pp. 153–167.

Eshelman, L. J., and Schaffer, J. D. (1993), "Real-Coded Genetic Algorithms and Interval-Schematai," in *Foundation of Genetic Algorithms 2*, ed. G. J. E. Rawlins, San Mateo, CA: Morgan Kaufmann, pp. 187–202.

Gelman, A., Roberts, R. O., and Gilks, W. R. (1996), "Efficient Metropolis Jumping Rules," in *Bayesian Statistics 5*, eds. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, New York: Oxford University Press.

Geman, S., and Geman, D. (1984), "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 721–741.

Geyer, C. J. (1991), "Markov Chain Monte Carlo Maximum Likelihood," in *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface* ed. E. M. Keramigas, Fairfax, VA: Interface Foundation, pp. 156–163.

Geyer, C. J., and Thompson, E. A. (1995), "Annealing Markov Chain Monte Carlo With Applications to Pedigree Analysis," *Journal of American Statistical Association*, 90, 909–920.

Gilks, W. R., Roberts, G. O., and George, E. I. (1994), "Adaptive Direction Sampling," *The Statistician*, 43, 179–189.

Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, & Machine learning*, Reading, MA: Addison-Wesley.

Green, P. J. (1995), "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination," *Biometrika*, 82, 711–732.

Hastings, W. K. (1970), "Monte Carlo Sampling Methods Using Markov Chain and Their Applications," *Biometrika*, 57, 97–109.

Hill, T., O'Connor, M., and Remus, W. (1996), "Neural Network Models for Time Series Forecasts," *Management Science*, 42, 1082–1092.

Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press.

Hornik, K., Stinchcombe, M., and White, H. (1989), "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, 2, 359–366.

Hukushima, K., and Nemoto, K. (1996), "Exchange Monte Carlo Method and Application to Spin Glass Simulations," *Journal of the Physics Society of Japan*, 65, 1604–1608.

Kang, S. (1991), "An Investigation of the Use of Feedforward Neural Networks for Forecasting," doctoral dissertation, Kent State University.

Liang, F., and Wong, W. H. (1999), "Dynamic Weighting in Simulations of Spin Systems," *Physics Letters*, Ser. A, 252, 257–262.

——— (2000), "Evolutionary Monte Carlo Sampling: Applications to C_p Model Sampling and Change-point Problem," *Statistica Sinica*, 10, 317–342.

Liu, J. S., Liang F., and Wong, W. H. (2000), "The Use of Multiple-Try Method and Local Optimization in Metropolis Sampling," *Journal of American Statistical Association*, 94, 121–134.

MacKay, D. J. C. (1992), "A Practical Bayesian Framework for Backprop Networks," *Neural Computation*, 4, 448–472.

Marinari, E., and Parisi, G. (1992), "Simulated Tempering: A New Monte Carlo Scheme," *Europhysics Letters*, 19, 451–458.

Mehrotra, K., Mohan, C. K., and Ranka, S. (1996), *Elements of Artificial Neural Networks*, Cambridge, MA: MIT Press.

Meng, X., and Wong, W. H. (1996), "Simulating Ratios of Normalizing Constants via a Simple Identity: A Theoretical Exploration," *Statistica Sinica*, 6, 831–860.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953), "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, 21, 1087–1091.

Müller, P., and Insua, D. R. (1998), "Issues in Bayesian Analysis of Neural Network Models," *Neural Computation*, 10, 749–770.

Neal, R. M. (1993), "Bayesian Learning via Stochastic Dynamics," in *Advances in Neural Information Processing Systems 5*, eds. C. L. Giles, S. J. Hansn, and J. D. Cowan, San Francisco: Morgan Kaufmann.

——— (1996), *Bayesian Learning for Neural Networks*, New York: Springer-Verlag.

——— (1999), "Erroneous Results in 'Marginal Likelihood from the Gibbs Output'," *http://www.cs.utoronto.ca/~raford/*.

Ono, I., and Kobayashi, S. (1997), "A Real-Coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover," in *Proceedings of the Seventh International Conference on Genetic Algorithms*, ed. T. Bäck, San Francisco: Morgan Kaufmann, pp. 246–253.

Phillips, D. B., and Smith, A. F. M. (1995), "Bayesian Model Comparison via Jump Diffusions," in *Markov Chain Monte Carlo in Practice*, eds. W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, London: Chapman and Hall, pp. 215–239.

Postman, M., Huchra, J. P., and Geller, M. J. (1986), "Probes of Large-Scale Structures in the Corona Borealis Region," *The Astronomical Journal*, 92, 1238–1247.

Ritter, C., and Tanner, M. A. (1992), "Facilitating the Gibbs Sampler: The Gibbs Stopper and the Griddy-Gibbs Sampler," *Journal of the American Statistical Association*, 87, 861–868.

Roberts, G. O., and Gilks, W. R. (1994), "Convergence of Adaptive Direction Sampling," *Journal of Multivariate Analysis*, 49, 287–298.

Roeder, K. (1990), "Density Estimation With Confidence Sets Exemplified by Superclusters and Voids in Galaxies," *Journal of the American Statistical Association*, 85, 617–624.

Rumelhart, D., Hinton, G., and McClelland, J. (1986), "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing*, eds. D. Rumelhart and J. McClelland, Cambridge, MA: MIT Press, pp. 45–76.

White, H. (1992), *Artificial Neural Networks: Approximation and Learning Theory*, Cambridge, MA: Blackwell.

Wong, W. H., and Liang, F. (1997), "Dynamic Weighting in Monte Carlo and Optimization," *Proceedings of the National Academy of Sciences*, 94, 14220–14224.

Wright, A. H. (1991), "Genetic Algorithms for Real Parameter Optimization," in *Foundations of Genetic Algorithms 1*, ed. G. J. E. Rawlins, San Mateo, CA: Morgan Kaufmann, pp. 205–218.