Statistica Sinica **10**(2000), 317-342

EVOLUTIONARY MONTE CARLO: APPLICATIONS TO C_p MODEL SAMPLING AND CHANGE POINT PROBLEM

Faming Liang and Wing Hung Wong

The National University of Singapore and UCLA, 8118 Math Sciences

Abstract: Motivated by the success of genetic algorithms and simulated annealing in hard optimization problems, the authors propose a new Markov chain Monte Carlo (MCMC) algorithm called an evolutionary Monte Carlo algorithm. This algorithm has incorporated several attractive features of genetic algorithms and simulated annealing into the framework of MCMC. It works by simulating a population of Markov chains in parallel, where a different temperature is attached to each chain. The population is updated by mutation (Metropolis update), crossover (partial state swapping) and exchange operators (full state swapping). The algorithm is illustrated through examples of C_p-based model selection and change-point identification. The numerical results and the extensive comparisons show that evolutionary Monte Carlo is a promising approach for simulation and optimization.

Key words and phrases: Change-point identification, crossover, exchange, genetic algorithm, Markov chain Monte Carlo, metropolis algorithm, mutation, parallel tempering, regression variable selection, simulated annealing.

1. Introduction

Simulated annealing (Kirkpatrick, Gelatt and Vecchi (1983)) and genetic algorithms (Holland (1975)) have been increasingly recognized by scientists as powerful tools for difficult computational problems. They have been applied successfully to combinatorial optimization (Randelman and Grest (1986), Chatterjee, Carrera and Lynch (1996), VLSI design (Wong, Leong and Liu (1988), Lienig (1997), Cohoon, Hedge, Martin and Richards (1991)), protein folding (Patton, Punch and Goodman (1995), Unger and Moult (1993)) and machine learning (Aarts and Korst (1989), Goldberg (1989)).

Simulated annealing mimics an annealing process in which the temperature of a system is first raised to a high value, then decreased slowly to a low one. At each temperature level, the Metropolis-Hastings algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller and Teller (1953), Hastings (1970)) is run long enough for the system to reach equilibrium (Geman and Geman (1984)). The simulation at high temperatures provides more opportunities for the system to escape from local minima. The merit of the increasing temperature idea has been adopted by

the recently developed MCMC algorithms, e.g., simulated tempering (Marinari and Parisi (1992)), parallel tempering (Geyer (1991), Hukushima and Nemoto (1996) and dynamic weighting (Wong and Liang (1997)).

The genetic algorithm (Holland (1975)) mimics the process of natural evolution. This algorithm differs from simulated annealing in two respects. First, rather than making changes to a single solution, a population of solutions are evolved, and the results obtained in the earlier stage of a run can be used as a guideline for the later steps. In other words, the genetic algorithm has an ability to learn from its previous results. Second, in addition to making local changes to the solutions (via mutation), the solutions also undergo nonlocal operations–crossover operations. The crossover operation is the key to the power of the genetic algorithm. However, it also makes the system highly nonlinear and strongly interacting, and the behavior of the algorithm is very difficult to analyse theoretically. Nix and Vose (1991) have applied a Markov chain analysis to this problem, but the analysis is not based on a realistic genetic algorithm, and it is also difficult to apply in practice. Other attempts include Walsh function analysis (Goldberg (1990)) and the schemata theory (Goldberg (1989)). But these analyses only focus on characterizing the fitness function and miss the effect of the dynamic feature of the genetic algorithm and consequently some predictions of them are wrong (Grefenstette (1992)).

In this paper, we propose a new algorithm—evolutionary Monte Carlo (EMC). This algorithm has incorporated many attractive features of simulated annealing and genetic algorithms into a framework of Markov chain Monte Carlo (MCMC). It works by simulating a population of Markov chains in parallel, where a different temperature is attached to each chain. The population is updated by mutation (Metropolis update in one single chain), crossover (partial states swapping between different chains), and exchange operators(full state swapping between different chains). The numerical results show that EMC offers a significant improvement over the traditional MCMC algorithms in both simulation and optimization.

Before the description of EMC, we have to mention that a similar idea, trying to use the population information to improve the efficiency of MCMC, was also pursued by Gilks, Roberts and George (1994). In their snooker algorithm, the Markov chain state is augmented to include a population of states, and each state is updated by a Gibbs sampler taken along a direction determined by some or all of the other states of the population. Quite recently, Liu, Liang and Wong (1998) suggested the sampling direction be determined by a local optimization procedure.

This paper is organized as follows. In Section 2 the genetic algorithm is briefly reviewed. Section 3 describes how the genetic algorithm is modified to

give a MCMC sampler. Section 4 presents some of the computational results, including the comparison with parallel tempering and the Gibbs sampler. Section 5 concludes the paper with a brief discussion.

2. Genetic Algorithm

In the natural evolutionary process, individuals in a population compete and mate with each other in order to produce increasingly stronger individuals. The genetic algorithm (Holland (1975)) mimics such a process and works as a general optimization technique in which the potential solutions to an optimization problem also undergo some genetic-styled operations in order to produce some that solutions minimize the objective function. Successful applications in many problems (Goldberg (1989)) show the usefulness of the algorithm.

For a clear description, let us first define some terms which here may have different meanings from their original biology connotations. Suppose we have an optimization problem, minimizing a cost function $H(x)$, where x may be a vector. In the terminology of genetic algorithms, $H(x)$ is called a "fitness" function, a possible solution x to the problem is referred to as an individual, and is represented by a vector $x = (\beta_1, \ldots, \beta_d)$ called a chromosome. A set of individuals is a population and the number of individuals in the population is the population size. A position, say i, on a chromosome is called a locus, β_i is called a gene, and the value of β_i is called a genotype. In most applications of genetic algorithms, the chromosome is coded as a binary vector with $\beta_i \in \{0, 1\}$. The population is updated by three genetic-styled operators: selection, crossover and mutation.

In the selection operator, some individuals are chosen from the current population according to a selection procedure to form a sub-population, called a mating population. The selection procedure most often used is the "roulette wheel" selection, in which each individual is assigned a weight, and is then selected with a probability proportional to weight. There are many methods for assigning weights, for example, each individual is assigned a weight proportional to its "Boltzmann probability",

$$p(x_i) = \frac{\exp(-H(x_i)/t)}{Z}, \qquad Z = \sum_{i=1}^{N} \exp(-H(x_i)/t). \tag{1}$$

Other selection procedures include random selection and truncation selection. In random selection, each individual is selected at random (the fitness values are not considered). In truncation selection, only the top p individuals (in fitness values) are selected.

In the crossover operator, different offspring are produced by a recombination of parental chromosomes randomly selected from the mating population. For

example, x_a and x_b are selected as the parental chromosomes and two "offspring" x'_a and x'_b are generated as follows. First an integer crossover point i is drawn uniformly on $\{1, \ldots, d\}$, then x'_a and x'_b are constructed by swapping the genes of the two parental chromosomes to the right of the crossover point. The following diagram shows the 1-point crossover operator.

$$\begin{array}{cc} (\beta_1^a, \ldots, \beta_d^a) & (\beta_1^a, \ldots, \beta_i^a, \beta_{i+1}^b, \ldots, \beta_d^b) \\ \Longrightarrow & \\ (\beta_1^b, \ldots, \beta_d^b) & (\beta_1^b, \ldots, \beta_i^b, \beta_{i+1}^a, \ldots, \beta_d^a). \end{array}$$

If there are k ($k > 1$) crossover points, 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 the position) of x'_a is randomly chosen from the two parental genotypes and the corresponding genotype of x'_b is assigned to the parental genotype not chosen by x'_a. The crossover operation is the key to genetic algorithms. With it, beneficial genes of parental chromosomes can combine together and possibly produce high quality individuals.

In the mutation operator, the individual chromosome is modified by means of point mutation. For example, if the chromosome is represented by a binary bit string, mutation can take place on a bit, either by reversing (i.e., change 0 to 1 and vice versa) or by throwing a coin for choosing 0 or 1 independently of the original value. Note that the bit is also chosen randomly. Often mutation operators are 1-point or 2-point mutations, i.e., one or two bits are chosen to mutate.

With the terms defined above, the genetic algorithm is summarized as follows.

1. Initialization. Set $t = 0$, start from the initial population $P(t)$ consisting of n individuals.
2. Selection. Set $t \leftarrow t + 1$. Some individuals are selected from the current population according to a selection procedure.
3. Crossover. Offspring are produced by a recombination of the chromosomes in the mating population. The offspring are denoted by $M(t)$ and new population $P(t)$ is formed by a selection from the mixed population of $P(t-1)$ and $M(t)$. Here the selection procedure may or may not be the same as in step 2.
4. Mutation. Choose offspring with a fixed and small probability p_m for further modifications, where p_m is called a mutation rate.
5. Termination checking. If the termination criterion is satisfied, stop; otherwise go back to step 2.

This algorithm has three free parameters: population size, mating population size and mutation rate. Population size may vary with problems from several tens

to several thousands; the mating population size is often chosen as 0.4 to 0.6 of the population; the mutation rate is usually quite small, e.g., 0.001 or 0.005. Refer to Davis (1991) for a detailed discussion of the choice of free parameters.

3. Evolutionary Monte Carlo

The genetic algorithm makes efficient use of information distributed across states of a population via a crossover operator, success in many hard optimization problems establishing usefulness. It is natural then to investigate whether the crossover operator can be used to improve the efficiency of a simulation. However, the genetic algorithm is an optimization technique and some modifications are needed to incorporate it into the framework of MCMC.

Let $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ denote a population of samples, where $x_i = (\beta_1^i, \ldots, \beta_d^i)$ is a d-dimensional vector called an individual or a chromosome in EMC, and N is the population size. In Bayesian statistics, x_i is often a vector of parameters while the fitness function $H(x_i)$ is the negative of the log-posterior of x_i. In EMC, a different temperature t_i is attached to each individual x_i, and the temperatures form a ladder with the ordering $t_1 > \cdots > t_N$. For convenience, $\boldsymbol{t} = (t_1, \ldots, t_N)$. Then a Boltzmann distribution for each individual x_i can be defined as

$$f_i(x_i) = \frac{1}{Z_i(t_i)} \exp\{-H(x_i)/t_i\}, \tag{2}$$

where $Z_i(t_i)$ is the normalizing constant, $Z_i(t_i) = \sum_{\{x_i\}} \exp\{-H(x_i)/t_i\}$.

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

$$f(\boldsymbol{x}) = \prod_{i=1}^{n} f_i(x_i) = \frac{1}{Z(\boldsymbol{t})} \exp\{-\sum_{i=1}^{N} H(x_i)/t_i\}, \tag{3}$$

where $Z(\boldsymbol{t}) = \prod_{i=1}^{N} Z_i(t_i)$.

In equation (3), a mutual independence is assumed for each individual. The choice for the distribution is partly motivated by parallel tempering (Geyer (1991), Hukushima and Nemoto (1996)), where such a distribution on an ensemble of states at various temperatures is used to facilitate a faster mixing. We now discuss how many aspects of the genetic algorithm can be incorporated into the framework of MCMC.

3.1. Mutation

In the mutation operator, a chromosome, say x_k, is randomly selected 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 which are also chosen randomly.

A new population is proposed as $\boldsymbol{y} = \{x_1, \ldots, y_k, \ldots, x_N\}$, and it is accepted with probability $\min(1, r_m)$ according to the Metropolis rule, where r_m is the Metropolis-Hastings ratio,

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

and $T(\cdot|\cdot)$ denotes the transition probability between populations. If the proposal is accepted, the current population $\boldsymbol{x}$ is replaced by $\boldsymbol{y}$, otherwise the population $\boldsymbol{x}$ is unchanged.

In addition to 1-point and 2-point mutations, we also use a uniform mutation in which each bit of x_k has a nonzero probability of mutating. Note that all these operators are symmetric, i.e., the transition probability from $\boldsymbol{x}$ to $\boldsymbol{y}$ is the same as that from $\boldsymbol{y}$ to $\boldsymbol{x}$.

3.2. Crossover

One chromosome pair, say x_i and x_j $(i \neq j)$, is selected from the current population $\boldsymbol{x}$ according to some selection procedure, e.g., a roulette wheel selection or a random selection. Without loss of generality we assume $H(x_i) \geq H(x_j)$. Two "offspring" are generated according to some crossover operator (discussed below), the offspring with a smaller fitness value is denoted y_j and the other is y_i. A new population is proposed as $\boldsymbol{y} = \{x_1, \ldots, y_i, \ldots, y_j, \ldots, x_N\}$. According to the Metropolis rule, the new population is accepted with probability $\min(1,r_c)$,

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

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 $\boldsymbol{x}$, $P((y_i, y_j)|(x_i, x_j))$ denotes the generating probability of (y_i, y_j) from the parental chromosomes (x_i, x_j).

Throughout this paper, the parental chromosomes are chosen as follows. The first chromosome x_i (x_j) is chosen according to a roulette wheel procedure with Boltzmann weights, i.e., x_i (x_j) is chosen with probability defined in (1). The second chromosome x_j (x_i) is chosen randomly from the rest of the population. The selection probability of (x_i, x_j) is then

$$P((x_i, x_j)|\boldsymbol{x}) = \frac{1}{(N-1)Z(\boldsymbol{x})}[\exp\{-H(x_i)/t\} + \exp\{-H(x_j)/t\}], \tag{6}$$

where $Z(\boldsymbol{x}) = \sum_{i=1}^{N} \exp\{-H(x_i)/t\}$. $P((y_i, y_j)|\boldsymbol{y})$ can be calculated similarly.

The crossover operators used in the genetic algorithm, e.g., 1-point crossover, 2-point crossover and uniform crossover, are also applicable here. Note that these

operators are all symmetric, i.e., $P((y_i, y_j)|(x_i, x_j))$=$P((x_i, x_j)|(y_i, y_j))$, the ratio of transition probabilities in (5) is reduced to the ratio of selection probabilities.

We also introduce a new crossover operator, which we refer to as an adaptive crossover. Here two offspring are generated as follows. For each locus, if x_i and x_j have the same value, y_i and y_j copy the value, and independently reverse it with probability p_0; if x_i and x_j have different values, y_i copies the value of x_i and reverses it with probability p_2, y_j copies the value of x_j and reverses it with probability p_1. Usually we set $0 < p_0 \le p_1 \le p_2 < 1$. Obviously, the adaptive crossover tends to preserve the good genotypes of a population, and learning is enhanced. Table 1 gives the single locus generating probabilities of new offspring (y_i, y_j). The generating probability of the new offspring can be written as

$$P((y_i, y_j)|(x_i, x_j)) = p_{11}^{n_{11}} p_{12}^{n_{12}} p_{21}^{n_{21}} p_{22}^{n_{22}}, \tag{7}$$

where p_{ab} and n_{ab} $(a, b = 1, 2)$ denote respectively the probability and frequency of the cell (a, b) of Table 1. Note that $n_{11} + n_{12} + n_{21} + n_{22} = d$.

Table 1. The single locus generating probabilities of new offspring.

parents: x_i and x_j	offspring: y_i and y_j	
	same	different
same	$p_0^2 + (1-p_0)^2$	$2p_0(1-p_0)$
different	$p_1(1-p_2) + p_2(1-p_1)$	$p_1p_2 + (1-p_1)(1-p_2)$

3.3. Exchange

This operation is the same as that introduced in parallel tempering (Geyer (1991)) and exchange Monte Carlo (Hukushima and Nemoto (1996)). Given the current population $\boldsymbol{x}$ and the temperature ladder $\boldsymbol{t}$, $(\boldsymbol{x}, \boldsymbol{t}) = (x_1, t_1, \ldots, x_N, t_N)$, we try to make an exchange between x_i and x_j without changing the t's, i.e., we try to change $(\boldsymbol{x}, \boldsymbol{t}) = (x_1, t_1, \ldots, x_i, t_i, \ldots, x_j, t_j, \ldots, x_N, t_N)$ to $(\boldsymbol{x}', \boldsymbol{t}) = (x_1, t_1, \ldots, x_j, t_i, \ldots, x_i, t_j, \ldots, x_N, t_N)$. The new population is accepted with probability min(1,r_e) according to the Metropolis rule, where

$$r_e = \frac{f(\boldsymbol{x}')}{f(\boldsymbol{x})} \frac{T(\boldsymbol{x}|\boldsymbol{x}')}{T(\boldsymbol{x}'|\boldsymbol{x})} = \exp\{(H(x_i) - H(x_j))(\frac{1}{t_i} - \frac{1}{t_j})\} \frac{T(\boldsymbol{x}|\boldsymbol{x}')}{T(\boldsymbol{x}'|\boldsymbol{x})}, \tag{8}$$

and $T(\cdot|\cdot)$ denotes the transition probability between populations. Typically, the exchange is only performed on states with neighboring temperature values, i.e., $|i - j| = 1$. Let $p(x_i)$ denote the probability that x_i is chosen to exchange with the other state, and $w(x_j|x_i)$ denote the probability that x_j is chosen to exchange with x_i for the given x_i. Thus, we have the transition probability $T(\boldsymbol{x}'|\boldsymbol{x}) = p(x_i)w(x_j|x_i) + p(x_j)w(x_i|x_j)$, and $T(\boldsymbol{x}'|\boldsymbol{x}) = T(\boldsymbol{x}|\boldsymbol{x}')$.

3.4. Algorithm

Based on the operators introduced above, EMC 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 of the Markov chain consists of two steps.

1. Apply the mutation or the crossover operator to the population with probabilities q_m and $1 - q_m$, respectively (q_m is the mutation rate).
2. Try to exchange x_i with x_j for N pairs (i, j) with i being sampled uniformly on $\{1, \ldots, N\}$ and $j = i \pm 1$ with probability $w(x_j|x_i)$, where $w(x_{i+1}|x_i) = w(x_{i-1}|x_i) = 0.5$ and $w(x_2|x_1) = w(x_{N-1}|x_N) = 1$.

In the mutation operation, each chromosome of the population is mutated independently. In the crossover operation, about 40% of chromosomes are chosen to mate. Note that the parental chromosomes are chosen in an iterative way, i.e., each time two parental chromosomes are chosen from the current population which has been updated by the last crossover operation. This operation repeats for $[N/5]$ (the integer part of $N/5$) times. The algorithm has three free parameters, namely N, $\boldsymbol{t}$ and q_m. The mutation rate q_m can be chosen to achieve a trade-off between the exploration and convergence of the algorithm (Spears (1992)). For a small population size, q_m is usually set to a large value to provide the system more opportunities to explore the sample space. However for a large population size, q_m is usually set to a small value to force the system to converge quickly. Typically, q is set to a value around 0.25 for a small or moderate population size (e.g, $N \leq 50$). The temperature ladder $\boldsymbol{t}$ can be set as in simulated tempering (Marinari and Parisi (1992)) and parallel tempering (Geyer (1991), Hukushima and Nemoto (1996)). Roughly speaking, t_i should be set such that

$$\mathrm{Var}(H(x_i))\delta^2 = O(1), \tag{9}$$

where $\delta = 1/t_{i+1} - 1/t_i$; $\mathrm{Var}(H(x_i))$ denotes the variance of $H(x_i)$, estimated through a preliminary run. This condition on δ is also equivalent to requiring that the distributions on temperature level t_i and t_{i+1} have a considerable overlap. Thus, we will have a reasonable acceptance probability for the exchange operations.

3.5. Parallel tempering

One important special case of EMC is parallel tempering. Setting $q_m = 1$, EMC is reduced to parallel tempering (Geyer (1991), Hukushima and Nemoto (1996)), of which each iteration consists of the following two steps.

- Update independently the N individuals using a standard MC method.
- Try to exchange x_i and x_j, the exchange being accepted with probability $\min(1, r_e)$ calculated in (8).

The first step can be accomplished by a mutation operator, the second step can be accomplished by the exchange operator. Parallel tempering and the closely related simulated tempering have been applied successfully in the simulation of spin-glasses systems (Marinari and Parisi (1992), Hukushima and Nemoto (1996)) and in Bayesian statistical inference (Geyer and Thompson (1995)). The successes establish their state-of-the-art positions as simulation algorithms in computational physics and statistics. The two algorithms have almost the same performance, but simulated tempering is difficult to implement due to the estimation of the pseudo-priors of the distributions on the temperature ladder.

4. Numerical Examples

4.1. Highway data

Weisberg (1985) used data to illustrate variable selection in a multiple regression. These data relate automobile accident rate (in accidents per million vehicle miles) to 13 potential independent variables, and including 39 sections of large highways in the state of Minnesota in 1973. According to the analysis of Weisberg (1985), variable 1 (the length of the segment in miles) should be included in the regression. Variables 11,12 and 13 are dummy variables that taken together indicate the type of highway, so one model should include either all or none of them. Thus, we regard them as one variable. After simplification, the problem has 10 independent variables, and the number of all possible subset models is reduced to 1024, manageable even for an exhaustive search. We use this example to test EMC as a simulation technique.

It is well known that Mallow's C_p statistic (Mallows (1973)) is an important measure in the regression variable selection. This is

$$C_p = \frac{RSS_p}{\hat{\sigma}^2} + 2p - n, \tag{10}$$

where p is the number of predictor variables included in the regression, RSS_p is the residual sum of squares from the subset model and $\hat{\sigma}^2$ is the estimated error variance calculated from the full model. Under the true model $E(C_p) = p$, so Mallows (1973) suggested that a good model have $C_p \cong p$. Since an exhaustive computation of C_p for all subset models is impossible for a problem with a large number of predictor variables, some methods have been proposed for the search of the minimum C_p, e.g., branch and bound (LaMotte and Hocking (1970)). Mallows himself has warned, however, that minimizing C_p may lead to the selection of a model that has a much larger MSEP (mean squared error of prediction) than the full model.

To illustrate EMC as a simulation approach, we apply it with a Boltzmann distribution defined on the space of all 1024 models. The Boltzmann distribution

is defined as

$$f_i(\mathbf{m}) = \frac{1}{Z(t_i)} \exp\{-C_p(\mathbf{m})/t_i\}, \tag{11}$$

where $\mathbf{m}$ denotes a model, t_i denotes a temperature, $C_p(\mathbf{m})$ denotes the C_p value of model $\mathbf{m}$, and $Z(t_i) = \sum_{\mathbf{m}} \exp\{-C_p(\mathbf{m})/t_i\}$. The interest in this distribution comes from the connection between Bayesian and non-Bayesian methods for variable selection in a linear regression. Liang and Wong (1999c) show that, with an appropriate prior setting, sampling from the posterior distribution of a linear regression model is approximately equivalent to sampling from the Boltzmann distribution defined in (11) with $t_i = 2$.

In this example, the chromosome is coded as a 10-dimensional binary vector; each component indicates whether the corresponding variable is included in the model or not. The highest temperature was set to 5, the lowest temperature was set to 1, the intermediate temperatures were equally spaced between the two limits. A 1-point mutation and a uniform crossover were used in the simulation, and the mutation rate was 0.25.

First, $f(\mathbf{m})$ was calculated for all 1024 models at temperature 1.0. Figure 2(a) shows this distribution (in terms of C_p) in the range $C_p < 10$, which covers 99.9% of the probability. In the first run, we chose a population size $N = 5$, the algorithm was run for 15000 iterations, and the CPU time was 105s (Here and in the next section, computations done on an Ultra Sparc2). The overall acceptance rates of mutation, crossover and exchange operations were 0.59, 0.60, and 0.76, respectively. These acceptance rates are quite high because the problem is very simple. During the run, 457 different C_p values were sampled at the lowest temperature 1.0. These included the smallest 331 C_p values and covered 99.9% probability of the target distribution. The histogram of the samples is shown in Figure 2(b). The figure shows that the Boltzmann distribution of C_p has been estimated accurately by the EMC samples. In another run the population size was set to 20, other parameters remaining the same. The algorithm was run for 10000 iteration with CPU 179s. The overall acceptance rates of mutation, crossover and exchange operations were 0.62, 0.51 and 0.95, respectively. The results are summarized in Figure 2. The convergence of EMC can be diagnosed using the Gelman-Rubin statistic $\hat{R}$ (Gelman and Rubin (1992)) based on multiple independent runs. Figure 1 shows $\hat{R}$ values computed from 10 independent runs. In this example, EMC converges quickly, usually within the first several tens of iterations ($\hat{R} < 1.1$). Here and elsewhere, we made a long run beyond convergence to get enough samples for inference tasks.

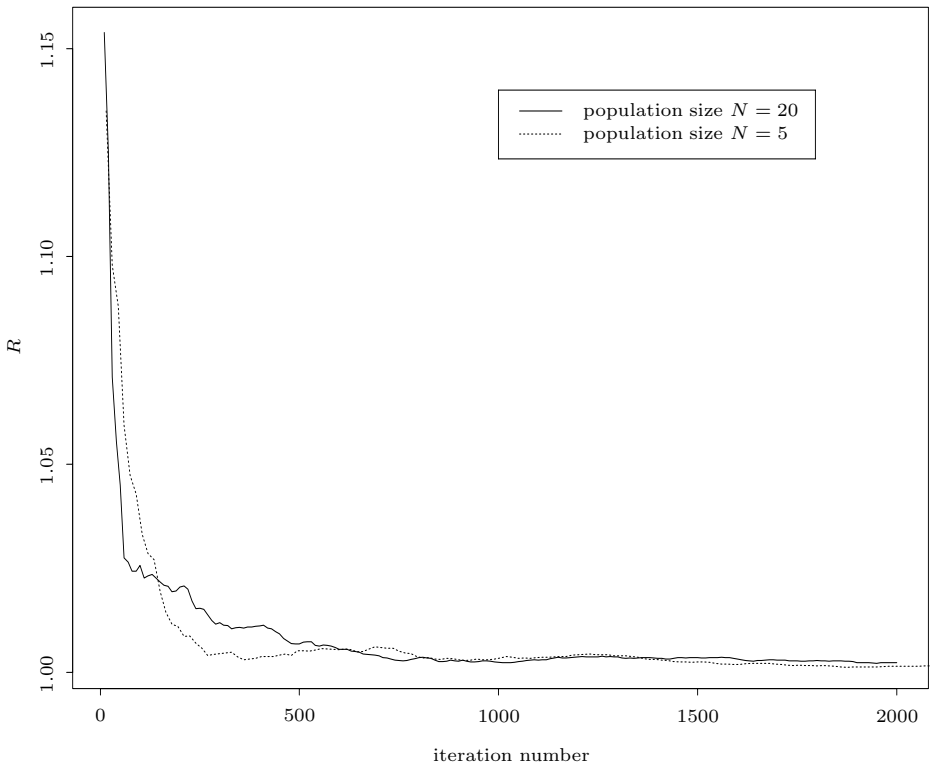

Figure 1. Gelman-Rubin convergence criterion $\hat{R}$. The $\hat{R}$ values are computed with 10 independent runs of EMC.

For comparison, parallel tempering was applied to the same example. A 1-point mutation was used as the local updating proposal function. We have also tried other proposal functions, e.g., 2-point mutation, and uniform mutation with similar results. In one run, the population size was set to 5 and 10000 iterations were produced within 105s. The overall acceptance rates of local updating and exchange operations were 0.59 and 0.77, respectively. Parallel tempering took more time per iteration, because in each iteration every chain (individual) was updated by one Metropolis-Hastings step. However, in the crossover step of EMC, only some (40%) of chromosomes were updated (chosen to produce offspring). In another run, the population size was 20, and 5000 iterations were produced within 193s. The overall acceptance rates of the local updating and exchange operations were 0.62 and 0.94, respectively. The results are summarized in Figure 2.

Figure 2(c) and 2(d) compare the convergence rates of parallel tempering and EMC for the population size 5 and 20 respectively, where "distance" is defined as the L^2 distance between the estimated mass vector and the true distribution

of C_p. In these figures, EMC and parallel tempering have been adjusted to have the same time scale. For example, in Figure 2(c), one point was plotted every 30 iterations for EMC, and one point was plotted every 20 iterations for parallel tempering to reflect computational time. The figures show that the EMC sampler has a faster convergence rate than parallel tempering. We also extended the run of parallel tempering to convergence, i.e., until the distance was lower than a threshold value. We found that the the computational time to convergence of parallel tempering was 3 times longer than EMC with $N = 5$ and with $N = 20$. In summary, EMC has sampled the Boltzmann distribution correctly and is more efficient than parallel tempering in this example.

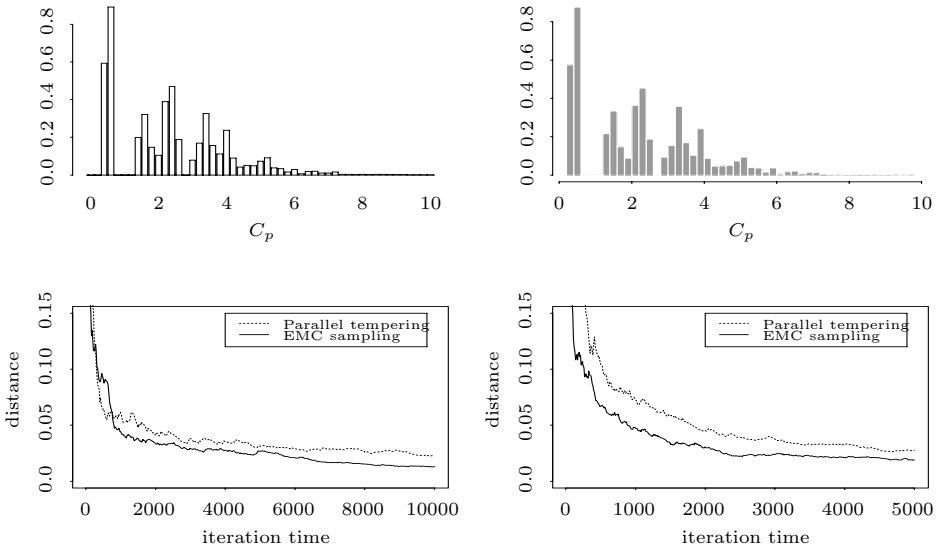

Figure 2. A comparison of EMC and parallel tempering for the highway example. (a) A bar graph of the Boltzmann distribution specified by C_p. The interval [0,10] was divided equally into 50 sub-intervals, each bar (the area of the bar) shows the mass value of the distribution in the sub-interval. (b) The histogram of C_p estimated by the samples from one run of EMC. (c) A comparison of the convergence rates of EMC and parallel tempering with the population size $N = 5$ and the same computational time. (d) A comparison of the convergence rates of EMC and parallel tempering with the population size $N = 20$ and the same computational time.

4.2. A hard model selection example

The minimum C_p for a problem with fewer than 30 predictor variables could be computed with the command *leaps*$(\cdot)$ in S-PLUS. This provides us a convenient way to test the ability of EMC in optimization. Note that the same type of

computational tasks also arise in Bayesian model selection (George and McCulloch (1993, 1995) Phillips and Smith (1995)). We use C_p values mainly because of our wish to compare the sampled smallest C_p value to the exact minimum.

We modified an example of George and McCulloch (1993) to generate the data for which the exact minimum C_p was computable in S-PLUS. In this example, 30 variables, $Y_1, \ldots, Y_{30}$, (each with length $n = 50$) were generated independently from the process $Y_i = Y_i^* + 2 * R$, where $Y_1^*, \ldots, Y_{30}^*$ i.i.d. $\sim N_{50}(0, 1)$ independent of $R \sim N_{50}(0, 1)$. This induced a pairwise correlation of (about) 0.67 among all independent variables. The dependent variable was generated according to the model $Z = (Y_1, \ldots, Y_{30})\beta + \epsilon$, where $\epsilon \sim N_{50}(0, \sigma^2 I)$ with $\sigma = 2$, and the coefficients $\beta = (\beta_1, \ldots, \beta_{30})'$ were chosen as $(\beta_1, \ldots, \beta_{10}) = (0, \ldots, 0)$, $(\beta_{11}, \ldots, \beta_{20}) = (1, \ldots, 1)$, $(\beta_{21}, \ldots, \beta_{30}) = (2, \ldots, 2)$. The number of total subset models is 2^{30}, thus it a hard example.

Here the chromosome was coded as a binary vector as in last example. The temperatures were equally spaced between 10 and 0.5. (We used a low temperature of 0.5, because we wanted to test whether the minimum C_p model could be sampled by EMC). The 1-point mutation and the adaptive crossover were used in the simulation with $p_m = 0.2$, $p_0 = 0.01$, $p_1 = 0.08$ and $p_2 = 0.1$. In the first run, the population size was $N = 5$, EMC was run for 1500 iterations and the CPU time was 60s. The overall acceptance rates of mutation, crossover and exchange operations were 0.38, 0.26, and 0.41, respectively. In the second run, the parameters had the same setting, but the population size was increased to 10. The algorithm was run for 750 iterations and the CPU time was 60s. The overall acceptance rates of mutation, crossover and exchange operations were 0.4, 0.27, and 0.64, respectively,

For comparison, parallel tempering was also applied to the example. With the same computational time, 800 and 400 iterations were produced in two runs with the population size $N = 5$ and $N = 10$, respectively. In both runs, the temperature setting was the same as EMC, uniform mutation was used as the proposal function, and each element was mutated with probability 0.09. In the run with $N = 5$, the overall acceptance probabilities of local updating and exchange were 0.21 and 0.42 respectively. In the run with $N = 10$, the overall acceptance probabilities of local updating and exchange were 0.23 and 0.65 respectively. The moderate acceptance rates of local updating suggest that parallel tempering has been implemented efficiently.

Table 2 summarizes the computational results of EMC and parallel tempering for 50 independent data sets. For each of them, the exact minimum C_p was computed by S-PLUS. The comparison shows that EMC offers a significant improvement over parallel tempering in locating the minimum C_p. For the failed cases in Table 2, EMC sampled minimum C_p's in other runs.

Table 2. A comparison of the results obtained by EMC and parallel tempering for 50 independent data sets, where "Error" denotes the number of data sets in which the minimum C_p was not sampled in the simulation.

	EMC		Parallel tempering	
	$N = 5$	$N = 10$	$N = 5$	$N = 10$
Error	1	2	14	27
Error rate	0.02	0.04	0.28	0.54

For one data set the EMC simulation was extended to 5500 iterations. The first 500 iterations were discarded for the burn-in process, and the remaining 5000 samples are shown in Figure 3. Figure 3(a) is the histogram of C_p sampled by EMC at the lowest temperature $t = 0.5$. The left most bar corresponds to the sampled minimum C_p, which is also the minimum C_p for the data set. Figure 3(b) is a bar graph of the frequencies with which each variable is included in the models sampled by EMC. This plot shows that the most frequently sampled variables include variables 11–14 and 16–30, and the sampling frequencies are higher than 0.6. A least squares regression of z on the variables 11–30 shows that variable 15 has the largest p-value 0.52 and the second largest p-value is only 0.09. This is again consistent with our simulation results.

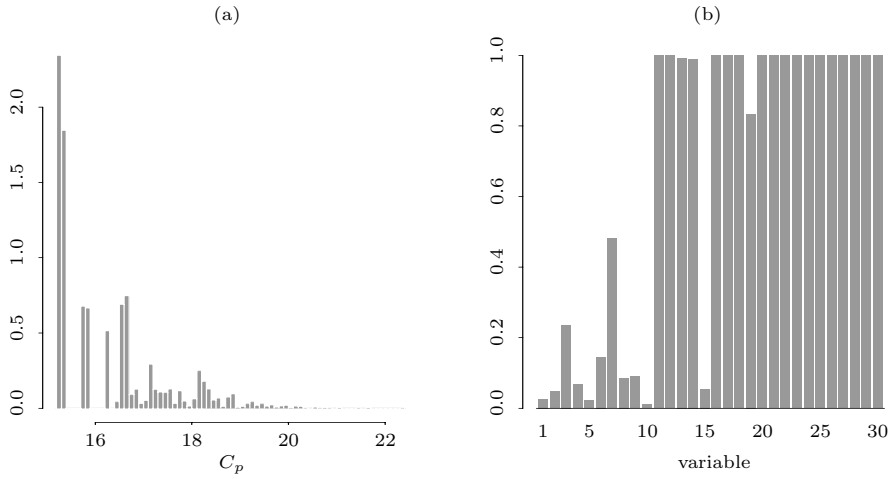

Figure 3. The results obtained by EMC in the C_p-based regression variable selection problem. (a) A histogram of C_p samples at the temperature 0.5; (b) A bar graph of the sampling frequencies of each variable.

Figure 4 shows the C_p ($C_p < 25$) values sampled by EMC in one run. It shows that the EMC simulation provides us a fruitful choice for model selection. In addition to the minimum C_p model (Model 1 with $C_p = 15.21$), many other

models with larger C_p values were also sampled: the true model (Model 2 with variables 11–30 and $C_p = 16.85$), a model with fewer variables (Model 3 with 18 variables and $C_p = 16.42$), and models near the line $C_p = p$ as suggested by Mallows (1973).

To illustrate further the power of EMC in simulation and optimization, we also compared the algorithm to the Gibbs sampler (Geman and Geman (1984)) and reversible jump MCMC (Green (1995)). The latter two algorithms are both widely used in statistics. For a fair comparison and a demonstration of the effectiveness of the crossover operator, the temperature ladder was set to a constant value in the following example.

We generated 30 independent variables, $Y_1,\ldots,Y_{15}, e_1,\ldots,e_{15}$ (each of length $n = 50$), where Y_i i.i.d. $\sim N_{50}(0,1)$, e_i *iid* $\sim N_{50}(0, 0.04)$. $Y_{16},\ldots,Y_{30}$ were constructed as $Y_i = Y_{i-15} + e_{i-15}$. The dependent variable was generated according to the model $Z = (Y_1,\ldots,Y_{30})\beta + \epsilon$, where $\epsilon \sim N_{50}(0,\sigma^2 I)$ with $\sigma = 2$, and the coefficients $\beta = (\beta_1,\ldots,\beta_{30})'$ were chosen as $(\beta_1,\ldots,\beta_{15}) = (0,\ldots,0)$, $(\beta_{16},\ldots,\beta_{30}) = (1,\ldots,1)$.

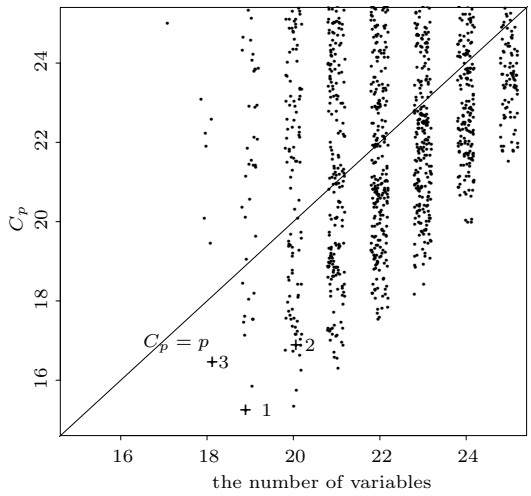

Figure 4. The scatter plot of C_p *vs* the number of variables included in the models sampled by EMC. Note that the points are horizontally jittered by adding random uniform (-.2,.2) numbers to the x-coordinates so the points are more readable. Point 1 denotes the model with minimum C_p ($C_p = 15.21$), point 2 denotes the true model ($C_p = 16.85$), point 3 denotes a model with 18 variables ($C_p = 16.42$).

As a consequence of the high correlation between certain pairs of independent variables, there are many local minima of C_p at a low temperature. The multi-

modality poses a real challenge. In the simulation of EMC, the temperature was set to a constant value of 0.25 and the population size was 50. The 1-point mutation and uniform crossover were used with $p_m = 0.2$. (This may not be the most effective setting.) For each data set, 500 iterations were simulated with CPU 150s. The overall acceptance rates of mutation and crossover operations were around 0.15 and 0.05, respectively. Within the same computational time and at the same temperature, the Gibbs sampler was run for 600 iterations (at each iteration, the components of each chromosome were scanned in a fixed order). In the second run, the setting was the same for all parameters, but the CPU time was extended to 1000 and 1200 iterations for EMC and Gibbs, respectively. The computational results are presented in Table 3 and Figure 5 (a) and (b).

The implementation of reversible jump MCMC is quite simple for the example, because it only involves jumps (i.e.,"birth/death" steps) in different dimensional spaces. Let Ω denote the set of all regression variables, i.e., $\Omega = \{Y_1, \ldots, Y_d\}$ and $d = 30$. Let S_k denote the current model which includes k variables. Given the current model S_k, with probability $b_k = 0.5$ $(k \neq 1, d)$, $b_1 = 1$ and $b_d = 0$, we propose to add ("birth") one variable to the regression, and with probability $1 - b_k$ we propose to delete ("death") one variable from the current regression. If we decide to add one variable to the regression, we pick one of the $d-k$ variables in $\Omega \setminus S_k$ for addition with equal probability $1/(d-k)$. Say variable Y_m is chosen for addition, then we change $k' = k + 1$ and set $S_{k'} = S_k + \{Y_m\}$. The proposal is accepted with probability $\min(1, A(birth))$, where

$$A(birth) = \exp(-(C_p(S_{k'}) - C_p(S_k))/t)\frac{1 - b_{k'}}{b_k}\frac{d-k}{k'}, \tag{12}$$

t is the temperature and $t = 0.25$. If we decide to delete one variable from the regression, we pick one of the k variables in S_k for deletion with equal probability $1/k$. Say the variable Y_m is chosen for deletion, then we change $k' = k - 1$ and set $S_{k'} = S_k - \{Y_m\}$. The proposal is accepted with probability $\min(1, A(death))$, where

$$A(death) = \exp(-(C_p(S_{k'}) - C_p(S_k))/t)\frac{b'_k}{1 - b_k}\frac{k}{d-k'}, \tag{13}$$

with $t = 0.25$. Note that the Jacobian term which appears in Green's (1995) formula does not appear here since we are proposing directly in the new parameter space. In the first run, the algorithm was run for 17500 iterations (one birth or death step is called one iteration) with the same CPU time (150s) as EMC. In the second run, with the same parameter setting, the algorithm was run for 35000 iterations. The results are presented in Table 3 and Figure 5 (c) and (d). In Figure 5, all histograms are left-skewed with respect to 0, which shows the superiority of EMC over the Gibbs sampler and RJMCMC in finding the

minimum C_p's in this example. EMC was also run with a temperature ladder for this example; the results were similar.

Clearly EMC provides a significant improvement over the Gibbs sampler and RJMCMC here. However, to show EMC has a better mixing rate is very difficult. A simple argument for the efficiency of EMC is given in the context of the C_p based model selection example.

Suppose we have one variable that should always be in the model. In the EMC algorithm, once this variable has been included in the model for one individual, it will not only stay there with high probability, it will tend to be included in the models of more and more individuals as the offspring of the original individual (in whom this variable appeared for the first time) multiply in the population. In contrast, the parallel Gibbs sampler, which gives equal chance for any variable to be included, must wait a much longer time for this variable to be present in most individuals. With regard to RJMCMC, here it reduces to a stepwise procedure consisting of "birth" and "death" steps. However, often it is desirable to have a proposal that incorporates both simultaneously, i.e., to exchange one variable in the current model by another outside the current model, especially one that has been found to improve Cp values. In order to achieve this, RJMCMC must proceed through a long series of single birth and death steps. In contrast, such an "exchange" operation can be achieved more efficiently in EMC through adaptive crossover. Finally, it is worth pointing out that, although EMC is still a Markov sampler, it is already endowed with certain learning capability. The reason is that the population can be viewed as a kind of memory that allows us to construct better proposals based on past observations, and the adaptive crossover operation is an effective means to exploit this memory.

Table 3. A comparison of the results obtained by the Gibbs sampler, reversible jump MCMC (RJMCMC) and EMC for 50 data sets, where $\#\{minimumC_p\}$ denotes the number of cases of which the exact minimum was sampled; "excess percentage" denotes the average excess of the sampled minimal C_p over the exact minimum, expressed as a percentage of the exact minimum.

Algorithm	run 1		run 2	
	$\#\{minimumC_p\}$ out of 50	excess percentage	$\#\{minimumC_p\}$ out of 50	excess percentage
EMC	27	1.63	40	0.63
Gibbs	17	4.61	24	2.47
RJMCMC	7	9.99	13	3.89

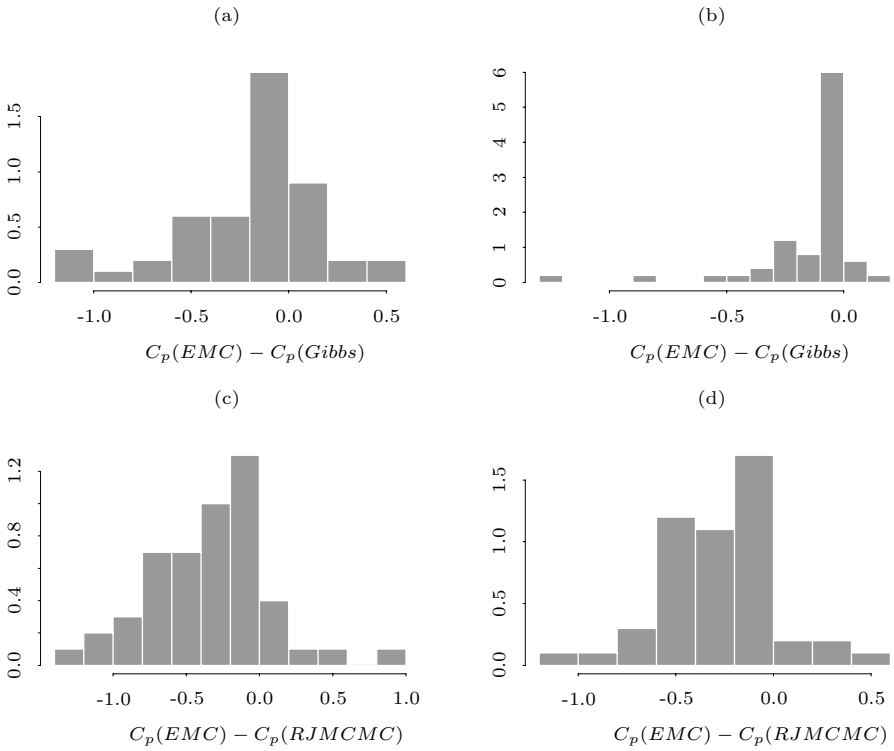

Figure 5. A comparison of results obtained by the Gibbs sampler, reversible jump MCMC (RJMCMC) and EMC for 50 data sets. Figures (a) and (b) show the histograms of the differences of the minimal C_p values sampled by EMC and Gibbs: (a) the first run; (b) the second run. Figure (c) and (d) show the histograms of the differences of the minimal C_p values sampled by EMC and RJMCMC: (c) the first run; (d) the second run.

4.3. Change-point identification

The problem is to identify the positions of an unknown number of change-points in a univariate time-series. Recently two posterior simulation-based methods, jump diffusion (Grenander and Miller (1994), Phillips and Smith (1995)) and reversible jump (Green (1995)), have been proposed for the problem. In this paper, we propose a different treatment. In our method, a latent vector is introduced to indicate the change-point positions, and the other parameters are integrated out with an appropriate choice of prior distributions. For simplicity, we assume that between change-points the observations are independently drawn from a Gaussian distribution $N(\mu, \sigma^2)$ with unknown μ and σ. After a change-point, the Gaussian distribution may shift in both mean and variance.

Let $Z = (z_1, \ldots, z_n)$ denote the observation sequence, in which we assume that there are no more than d change-points, e.g., $d = [n/2]$. A d-dimensional binary vector $\beta = (\beta_1, \ldots, \beta_d)$ is introduced as a latent vector and it works as a chromosome in EMC. A value 1 of β_i indicates that a change-point is identified at the ith position, otherwise no change-point is identified there. Let $\beta^{(k)}$ correspond to a model with k change points, with unknown positions of the change-points be denoted by $c_1, \ldots, c_k$, $c_1 < c_2 < \cdots < c_k$. For convenience, we let $c_0 = 1$ and $c_{k+1} = n$. Let μ_i and σ_i^2 denote the mean and variance of the Gaussian distribution between change-points c_{i-1} and c_i for $i = 1, \ldots, k+1$. Then the set of free parameters of model $\beta^{(k)}$ is $\theta^{(k)} = (\beta_1, \ldots, \beta_d, \mu_1, \sigma_1^{-2}, \ldots, \mu_{k+1}, \sigma_{k+1}^{-2}) = (\beta^{(k)}, \mu_1, \sigma_1^{-2}, \ldots, \mu_{k+1}, \sigma_{k+1}^{-2})$ and the total number of parameters is $n(k) = d + 2k + 2$. Let $\mathcal{M}_k$ denote the model space with k change-points with $\beta^{(k)} \in \mathcal{M}_k$, with $\Omega = \cup_{k=0}^{d} \mathcal{M}_k$.

The log-likelihood function of model $\beta^{(k)}$ is then

$$L(Z|\theta^{(k)}) = -\sum_{i=1}^{k+1}\{(c_i - c_{i-1}) \log \sigma_i + \frac{1}{2\sigma_i^2} \sum_{j=c_{i-1}+1}^{c_i} (z_j - \mu_i)^2\}. \tag{14}$$

The prior distributions are specified as follows. First, the prior probability of the model space $\mathcal{M}_k$ is specified by a truncated Poisson distribution with unknown parameter λ, and each of the $d!/[k!(d-k)!]$ models in $\mathcal{M}_k$ is *a priori* equally likely. Hence,

$$p(\beta^{(k)}) = \frac{\lambda^k}{\sum_{j=0}^{d} \frac{\lambda^j}{j!}} \frac{(d-k)!}{d!}, \qquad k = 0, 1, \ldots, d. \tag{15}$$

The parameters μ_i and σ_i^{-2} are assumed to be independent *a priori*, and an improper uniform prior is put on each μ_i, a $Gamma(\gamma, \delta)$ prior is put on each σ_i^{-2} with known γ and δ. Then the log-prior of $(\mu_1, \sigma_1^{-2}, \ldots, \mu_{k+1}, \sigma_{k+1}^{-2})$ is

$$\log p(\mu_1, \sigma_1^{-2}, \ldots, \mu_{k+1}, \sigma_{k+1}^{-2}) = a_k - \sum_{i=1}^{k+1}\{(\gamma - 1) \log \sigma_i^2 + \frac{\delta}{\sigma_i^2}\}, \tag{16}$$

where $a_k = (k+1)(\gamma \log \delta - \log \Gamma(\gamma))$. The log-posterior of θ can be obtained (up to an additive constant) by adding equations (14), (15) and (16).

Our aim is to sample from the marginal posterior $p(\beta|Z)$. Under the prior setting, it can be obtained by integrating out $\mu_1, \sigma_1^{-2}, \ldots, \mu_{k+1}, \sigma_{k+1}^{-2}$ explicitly from the full posterior $p(\theta|Z)$, i.e.,

$$p(\beta^{(k)}|Z) = \int \cdots \int p(\beta^{(k)}, \mu_1, \sigma_1^{-2}, \ldots, \mu_{k+1}, \sigma_{k+1}^{-2}|Z) d\mu_1 d\sigma_1^{-2} \cdots d\mu_{k+1} d\sigma_{k+1}^{-2}. \tag{17}$$

The resulting marginal posterior (in logarithm) of β is (up to an additive constant),

$$L(\beta^{(k)}|Z) = b_k - \sum_{i=1}^{k+1} \Big\{ \frac{1}{2}\log(c_i - c_{i-1}) - \log\Gamma(\frac{c_i - c_{i-1} - 1}{2} + \gamma) + (\frac{c_i - c_{i-1} - 1}{2} + \gamma)\log[\delta + \frac{1}{2}\sum_{j=c_{i-1}+1}^{c_i} z_j^2 - \frac{(\sum_{j=c_{i-1}+1}^{c_i} z_j)^2}{2(c_i - c_{i-1})}] \Big\}, \quad (18)$$

where $b_k = a_k + ((k+1)/2)\log 2\pi + \log(d-k)! + k\log\lambda$. The prior provides us a very simple marginal posterior for β, and the resulting posterior simulation can be done only in the model space.

In this example, the observation data consists of 200 observations with $z_1, \ldots, z_{50}$ i.i.d. $\sim N(-0.5, 1.0)$; $z_{51}, \ldots, z_{90}$ i.i.d. $\sim N(0.5, 0.16)$; $z_{91}, \ldots, z_{140}$ i.i.d. $\sim N(0, 1)$; and $z_{141}, \ldots, z_{200}$ i.i.d. $\sim N(0.75, 1.0)$. The time plot is shown in Figure 6.

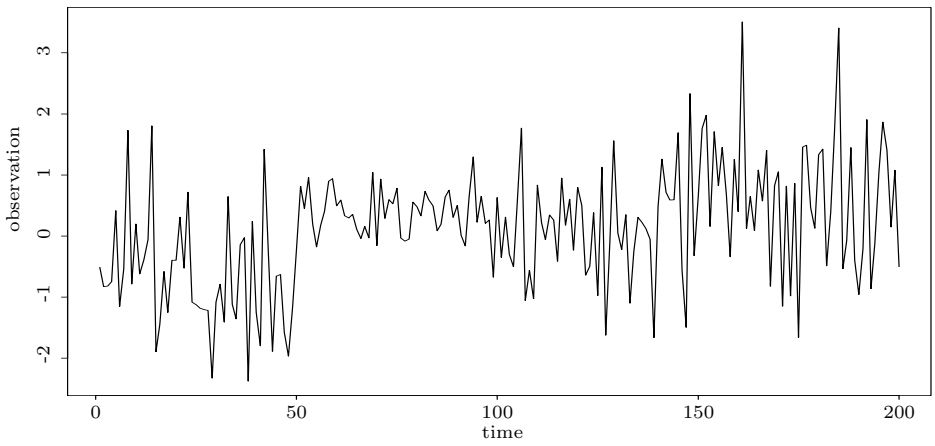

Figure 6. A time series plot of the observation data of the change-point identification example.

We assume that there are no more than 99 change-points in the observation sequence, and the change-points only occur after even observations, i.e., $d = 99$, and $c_i \in \{2, 4, \ldots, 198\}$. In the simulation, we set $\lambda = 3$, $\gamma = 0.05$ and $\delta = 0.05$, which corresponds to a vague prior on σ_i^{-2}. The population size was 20 and the temperature levels were equally spaced between 5 and 1. A three-point mutation and an adaptive crossover were used with $p_m = 0.2$, $p_0 = 0.01$, $p_1 = 0.02$ and $p_2 = 0.04$. EMC was run for 6000 iterations (the first 1000 iterations were discarded for the burn-in process), the CPU time was 127s on a workstation Alpha-500. The

overall acceptance probabilities of mutations, crossover and exchange operations were 0.38, 0.21 and 0.75, respectively. Table 4 lists the 15 models with the largest log-posterior values found by EMC. Note that the true change-point model was ranked 14 in log-posterior values among all sampled models. Other results of the run are shown in Figure 7.

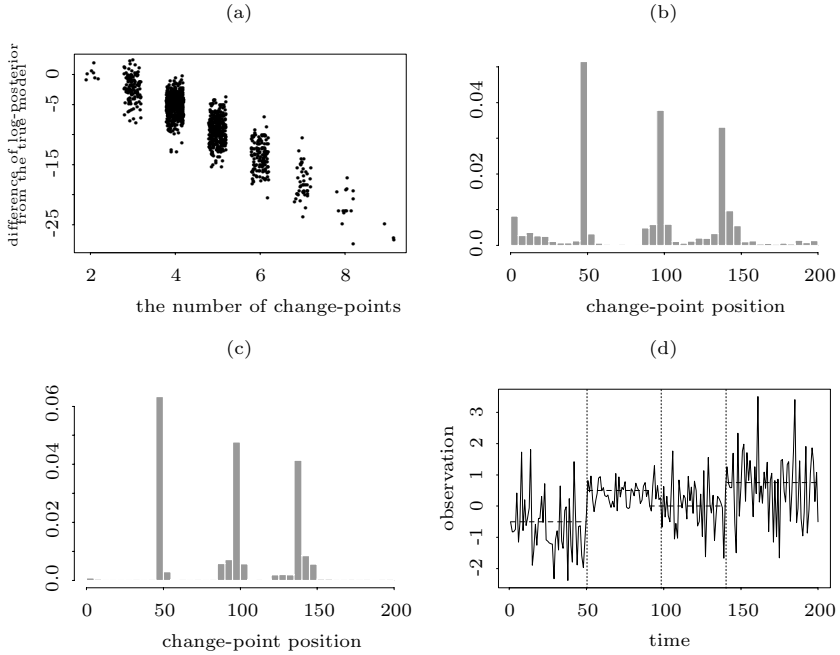

Figure 7. The simulation results of EMC for the change-point example: (a) The scatter plot of the difference of the log-posterior values from the true model *vs* the number of change-points. The points are horizontally jittered by adding random uniform (-.2,.2) numbers to the x-coordinates; (b) The posterior histogram of change-point positions sampled by EMC; (c) The posterior histogram of change-point positions conditional on the model space $\mathcal{M}_3$ sampled by EMC; (d) A comparison of the MAP estimate of the change-point positions and the true change-point positions (the vertical (dotted) lines indicated the change-point positions identified by the MAP model, the horizontal (dashed) lines indicate the change-point positions of the true model).

Figure 7(a) shows that the MAP (maximum *a posteriori*) model was sampled in $\mathcal{M}_3$. In addition, many models were sampled for which the log-posterior values were very close. Figure 7(b) is the posterior histogram of the identified change-point positions. It shows that most of the models with high posterior probabilities have 3 change-points. Figure 7(c) is the posterior histogram of change-point positions conditional on the model $\mathcal{M}_3$. This figure shows that the

three most likely change points are around 50, 100 and 140. Note that there is much uncertainty around the third cluster of the histogram bars in Figure 7(c). This is consistent with the results shown in Table 4: the models with the first two change points at 50 and 98, and the third change point being around 140 (134–146) have very close log-posterior values. Figure 7(d) shows the MAP estimate of the change patterns of the data. The MAP estimate of the change-points is (50,98,140), which differs from the true values (50, 90, 140) at the second change-point. This result is strongly supported by the data: a detailed data exploration shows that the observations 91-98 have a sample mean that is larger than expected. Also, the data have a great deal of noise, and it is very difficult to decide if an observation comes from one or the other of the two neighboring distributions. The EMC simulation seems to work well in identifying some plausible models according to the posterior likelihood.

Table 4. The 15 models with the largest log-posterior values sampled by EMC. The underlined model is true and is ranked 14 in log-posterior values among all models sampled by EMC. The second column shows the differences of the log-posterior values of the models from the true model.

No.	log-posterior*	number of change-points	change patterns
1	2.18	3	(50, 98, 140)
2	2.02	3	(50, 98, 138)
3	1.80	2	(50, 98)
4	1.60	3	(50, 98, 144)
5	1.34	3	(50, 98, 142)
6	0.98	3	(50, 98, 136)
7	0.87	3	(50, 96, 140)
8	0.85	3	(50, 98, 146)
9	0.34	2	(50, 102)
10	0.33	3	(50, 96, 144)
11	0.29	3	(50, 98, 134)
12	0.04	3	(50, 96, 142)
13	0.01	2	(50, 96)
14	0.00	3	(50, 90, 140)
15	−0.02	2	(50, 100)

5. Discussion

This paper introduces an evolutionary Monte Carlo algorithm. The algorithm works by simulating a population of Markov chains in parallel, where a different temperature is attached to each chain. The population is updated by mutation, crossover and exchange operators, and the updates are accepted or rejected according to the Metropolis rule.

The effectiveness of the algorithm is due to two things. First, the algorithm has incorporated the learning ability of the genetic algorithm by evolving with crossover operators. The "learning" or adaptive nature of the algorithm plays an important role in the simulation of EMC, especially in the early stage of the simulation. EMC works on a population of Markov chains. By chance, some sample (individual or chromosome) obtained in one chain is better than the others in fitness value. In subsequent crossover steps, according to the selection procedure employed by EMC, the individual with a higher fitness value will have a larger probability to be chosen as a parental chromosome to mate with the other parental chromosome to produce offspring by exchanging parts of their genomes. For example, x_a and x_b are chosen as parental chromosomes and x_a has a higher fitness value, with resulting offspring x'_a and x'_b. We can hope that x'_b will be more alike in genes to x_a than one obtained by some random changes on x_b, and there will also be a large probability for x'_b to have a higher fitness value than x_b. In this sense, x'_b has learned from x_a. The good samples obtained in some chains will guide the search in the further steps. In other words, the crossover operator allows us to construct better proposal distributions based on the information learned in the EMC process and stored in the population. This is also consistent with the argument of Gilks, Richardson and Spiegelhalter (1995), which points out that the rate of (Markov chain) convergence to the stationary distribution depends crucially on the relationship between the proposal function and the target distribution.

Second, the algorithm has incorporated the attractive feature of simulated annealing by sampling along a temperature ladder. The simulation at high temperature can help the system escape from local minima, and this substantially accelerates the mixing of the system. It is known that selection is an essential ingredient in all evolutionary processes. In this connection, we note that the exchange operation (swapping of temperature) can be viewed as a selection mechanism. An individual with poor fitness will, through the exchange operations, be forced to climb up the temperature ladder. At high temperature levels, random mutations are easily accepted and thus the gene of this poor-fitness individual will be eliminated from the population.

Although the algorithm was developed for sampling from a space of binary sequences, it is already applicable in many areas. In addition to the statistical model selection problem (George and McCulloch (1993), Brown and Vannucci (1998)) and the change-point identification problem, the algorithm can also find applications in physics and technology. Examples include the simulation of the Ising model (Swedsen and Wang (1987), Liang and Wong (1999a)), combinatorial optimization (Aarts and Korst (1989)), protein folding on lattices, and VLSI design. In all these problems, chromosomes can be encoded by binary vectors.

The generalization to problems with a sample space of finite alphabet sequences is also immediate. One simple method is to code the finite alphabet sequence by a finite binary sequence. Another method modifies the mutation and crossover operators such that each locus of the chromosomes can take values in the entire alphabet set. The algorithm has also been generalized to real parameter problems in Liang and Wong (1999b), in which each chromosome is coded by a real parameter vector. All computational results show the effectiveness of the crossover operator. Further work on the theory of EMC is of interest.

References

Aarts, E. and Korst, J. (1989). *Simulated Annealing and Boltzmann Machine.* John Wiley, New York.

Brown, P. J. and Vannucci, M. (1998). Multivariate Bayesian variable selection and prediction. *J. Roy. Statist. Soc. Ser. B* **60**, 627-641.

Chatterjee, S., Carrera, C. and Lynch, L. A. (1996). Genetic algorithms and traveling salesman problems. *European J. Operational Research* **93**, 490-510.

Cohoon, J. P., Hedge, S. U., Martin, W. N. and Richards, D. (1991). Distributed genetic algorithms for the floorplan design problem. *IEEE Trans. on CAD* **10**, 484-492.

Davis, L. (1991). *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences (with discussion). *Statist. Sci.* **7**, 457-472.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. and Machine Intelligence* **6**, 721-741.

George, E. I. and McCulloch, R. E. (1993). Variable selection via Gibbs sampling. *J. Amer. Statist. Assoc.* **88**, 881-889.

George, E. I. and MiCulloch, R. E. (1995). Stochastic search variable selection. *Markov Chain Monte Carlo in Practice* (Edited by W. R. Gilks, S. Richardson and D. J. Spiegelhalter), 203-214. Chapman & Hall, London.

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proceedings of the* 23*rd Symposium on the Interface* (Edited by E. M. Keramigas), 156-163. Interface Foundations, Fairfax.

Geyer, C. J. and Thompson, E. A. (1995). Annealing Markov chain Monte Carlo with applications to pedigree analysis. *J. Amer. Statist. Assoc.* **90**, 909-920.

Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. (1995). Introducing Markov chain Monte Carlo. In *Markov Chain Monte Carlo in Practice* (Edited by W. R. Gilks, S. Richardson and D. J. Spiegelhalter), 1-19. Chapman & Hall, London.

Gilks, W. R., Roberts, G. O. and George, E. I. (1994). Adaptive direction sampling. *Statistician* **43**, 179-189.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley.

Goldberg, D. E. (1990). Genetic algorithms and Walsh functions: part I, a gentle introduction. *Complex Systems* **3**, 129-152.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* **82**, 711-732.

Grefenstette, J. J. (1992). Deception considered harmful. In *Foundations of Genetic Algorithms 2* (Edited by L. Darrel Whitley). Morgan Kaufmann, San Mateo.

Grenander, U. and Miller, M. I. (1994). Representations of knowledge in complex systems. *J. Roy. Statist. Soc. Ser. B* **56**, 549-603.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**, 97-109.

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

Hukushima, K. and Nemoto, K. (1996). Exchange Monte Carlo method and application to spin glass simulations. *J. Phys. Soc. Jpn.* **65**, 1604-1608.

Kirkpatrick, S., Gelatt, Jr., C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science* **220**, 671-680.

LaMotte, L. R. and Hocking, R. R. (1970). Computational efficiency in the selection of regression variables. *Technometrics* **12**, 83-93.

Liang, F. and Wong, W. H. (1999a). Dynamic weighting in simulations of spin systems. *Phys. Lett. A* **252**, 257-262.

Liang, F. and Wong, W. H. (1999b). Real parameter evolutionary Monte Carlo with applications in Bayesian neural networks. Technical Report, Department of Statistics and Applied Probability, NUS.

Liang, F. and Wong, W. H. (1999c). Automatic Bayesian variable selection in linear selection and the related non-Bayesian methods. Technical Report, Department of Statistics and Applied Probability, NUS.

Lienig, L. (1997). A parallel genetic algorithm for performance-driven VLSI routing. *IEEE Trans. on Evolutionary Comp.* **1**, 29-39.

Liu, J. S., Liang, F. and Wong, W. H. (1998). The use of multi-try method and local optimization in Metropolis sampling. *J. Amer. Statist. Assoc.*, (in press).

Mallows, C. L. (1973). Some comments on C_p. *Technometrics* **15**, 661-676.

Marinari, E. and Parisi, G. (1992). Simulated tempering: a new Monte Carlo scheme. *Europhys. Lett.* **19**, 451-458.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087-1091.

Nix, A. and Vose, M. D. (1991). Modeling genetic algorithms with Markov chains. *Ann. Math. Artificial Intelligence* **5**, 79-88.

Patton, A. L., Punch III, W. F. and Goodman, E. D. (1995). A standard GA approach to native protein conformation prediction. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 574-581.

Phillips, D. B. and Smith, A. F. M. (1995). Bayesian model comparison via jump diffusions. In *Markov Chain Monte Carlo in Practice* (Edited W. R. Gilks, S. Richardson and D. J. Spiegelhalter), 215-239. Chapman & Hall, London.

Randelman, R. E. and Grest, G. S. (1986). N-city traveling salesman problem: optimization by simulated annealing. *J. Statist. Phys.* **45**, 885-890.

Spears, W. M. (1992). Crossover or mutation? In *Foundations of Genetic Algorithms 2* (Edited by L. D. Whitley). Morgan Kaufmann, San Mateo.

Swedsen, R. H. and Wang, J. S. (1987). Nonuniversal critical dynamics in Monte Carlo simulations. *Phys. Rev. Lett.* **58**, 86-88.

Unger, R. and Moult, J. (1993). Genetic algorithms for protein folding simulations. *J. Molecular Biology* **231**, 75-81.

Weisberg, S. (1985). *Applied Linear Regression*. John Wiley, New York.

Wong, D. F., Leong, H. W. and Liu, C. L. (1988). *Simulated Annealing for VLSI design*. Kluwer, Boston.

Wong, W. H. and Liang, F. (1997). Dynamic weighting in Monte Carlo and optimization. *Proc. Natl. Acad. Sci. USA* **94**, 14220-14224.

Wright, A. H. (1991). Genetic algorithms for real parameter optimization. In *Foundations of Genetic Algorithms* (Edited by G. J. E. Rawlins), 205-218. Morgan Kaufmann, San Mateo.

Department of Statistics and Applied Probability, National University of Singapore, 3 Science Drive 2, Singapore 117543.

E-mail: stalfm@nus.edu.sg

Department of Statistics, UCLA, 8118 Math Sciences, 405 Hilgard Ave., Los Angeles, CA 90095, U.S.A.

E-mail: whwong@stat.ucla.edu

(Received May 1999; accepted December 1999)