Journal of Biometrics & Biostatistics
ISSN: 2155-6180

Wu and Liang, J Biomet Biostat 2011, S1
DOI: 10.4172/2155-6180.S1-002

Research Article Open Access

Population SAMC vs SAMC: Convergence and Applications to Gene Selection Problems

Mingqi Wu[1] and Faming Liang[2]*

[1]Mingqi Wu, Merck Research Laboratories, North Wales, PA
[2]Faming Liang, Department of Statistics, Texas A&M University, College Station, Texas

Abstract

The Bayesian model selection approach has been adopted by more and more people when analyzing a large data. However, it is known that the reversible jump MCMC (RJMCMC) algorithm, which is perhaps the most popular MCMC algorithm for Bayesian model selection, is prone to get trapped into local modes when the model space is complex. The stochastic approximation Monte Carlo (SAMC) algorithm essentially overcomes the local trap problem suffered by conventional MCMC algorithms by introducing a self-adjusting mechanism based on the past samples. In this paper, we propose a population SAMC (Pop-SAMC) algorithm, which works on a population of SAMC chains and can make use of crossover operators from genetic algorithms to further improve its efficiency. Under mild conditions, we show the convergence of this algorithm. Comparing to the single chain SAMC algorithm, Pop-SAMC provides a more efficient self-adjusting mechanism and thus can converge faster. The effectiveness of Pop-SAMC for Bayesian model selection problems is examined through a change-point identification problem and a gene selection problem. The numerical results indicate that Pop-SAMC significantly outperforms both the single chain SAMC and RJMCMC.

Introduction

Given data, how to select an optimal model, according to some criteria, from a set of possible models is an important topic for statistician. Under the Bayesian framework, the success of choosing the right model relies on how accurately one can estimate the posterior probability of each of the possible models. Although the reversible jump MCMC (RJMCMC) algorithm [1] can work well for many Bayesian model selection problems for which the model space is simple, it is prone to get trapped into local optimal models when the model space is complex, i.e., consisting of a multitude of models separated by high energy barriers.

To overcome the local-trap problem, [2] proposed a stochastic approximation Monte Carlo (SAMC) algorithm. The basic idea of SAMC can be described as follows. Suppose that we are interested in sampling from a distribution,

$$f(x) = c\psi(x), \quad x \in X, \tag{1}$$

where X is the sample space and c is an unknown constant. Let $E1,\ldots,Ek$ denote a partition of X, and let $wi = \int_{Ei} \psi(x)dx$ for i=1,…,k. SAMC seeks to draw samples from the trial distribution

$$f_w(x) \propto \sum_{i=1}^{k} \frac{\pi_i \psi(x)}{w_i} I(x \in E_i) \tag{2}$$

Where π_i's are pre-specified constants such that $\pi_i > 0$ for all i and $\sum_{i=1}^{k} \pi_i = 1$, which define the desired sampling frequency for each of the subregions. If $w_1,\ldots,w_k$ are known, sampling from $f_w(x)$ will result in a "random walk" in the space of subregions (by regarding each subregion as a point) with each subregion being sampled with a frequency proportional to π_i. Hence the local-trap problem can be overcome essentially, provided that the sample space is partitioned appropriately. The way to partition sample space is problem dependent. For example, if one's goal is to maximize the target distribution, then one can partition the sample space according to the density function; if one's goal is at model selection, then one may partition the sample space according to the index of models.

The success of SAMC depends on whether w_i's can be well estimated. SAMC provides a systematic way to estimate wi in an online manner. Let θ_{ti} denote the working estimate of $\log(wi/\pi_i)$ obtained at iteration t, and let $\theta t = (\theta t1,\ldots, \theta tk) \in \Theta$, where Θ denotes a compact set. Let $\{\gamma_t\}$ be a positive, nondecreasing sequence satisfying

$$(i)\sum_{t=1}^{\infty} \gamma_t = \infty, \ (ii)\sum_{t=1}^{\infty} \gamma_t^{\zeta} < \infty, \tag{3}$$

for any $\zeta > 1$. For example, one may set

$$\gamma_t = \frac{T_0}{\max(T_0, t)}, \qquad t = 1,2,\cdots \tag{4}$$

for some value $T_0 > 1$. Under the above setting, one iteration of SAMC consists of the following steps:

The SAMC algorithm:

Metropolis-Hastings (MH) sampling Simulate a sample χ_t by a single MH update with the invariant distribution

$$f_{\theta_t}(x) \propto \sum_{i=1}^{k} \frac{\psi(x)}{e^{\theta_{ti}}} I(x \in E_i) \tag{5}$$

Weight updating Set

$$\theta^* = \theta_t + \gamma_{t+1}(et - \pi) \tag{6}$$

where $e_t = (I(x_t \in E_1),\ldots,I(x_t \in E_k))$ and $I(\cdot)$ is the indicator function. If $\theta^* \in \Theta$, set $\theta_{t+1} = \theta^*$; otherwise, set $\theta_{t+1} = \theta^* + C^*$, where $C^* = (c^*,L,c^*)$ can be an arbitrary vector which satisfies the condition $\theta^* + C^* \in \Theta$. Note that $f_\theta(x)$ is invariant to this location transformation of θ^*.

***Corresponding author:** Faming Liang, Department of Statistics, Texas A&M University, College Station, Texas, E-mail: fliang@stat.tamu.edu

Received June 15, 2011; **Accepted** November 01, 2011; **Published** November 04, 2011

Citation: Wu M, Liang F (2011) Population SAMC vs SAMC: Convergence and Applications to Gene Selection Problems. J Biomet Biostat S1:002. doi:10.4172/2155-6180.S1-002

A remarkable feature of SAMC is its self-adjusting mechanism, which operates based on past samples. This mechanism penalizes the over-visited subregions and rewards the under-visited subregions, and thus enables the system to escape from local traps very quickly. Mathematically, if a subregion j is visited at time t,$\theta_{t+1,j}$ will be updated to a larger value,$\theta_{t+1,i} \leftarrow \theta_{t+1} + \gamma_{t+1}(1-\pi_i)$, such that, this subregion has a smaller probability to be visited in the next iteration. On the other hand, for those regions, $j(j \neq i)$, not visited this iteration, $\theta_{t+1,j}$ will decrease to a smaller value, $\theta_{t+1,j} \leftarrow \theta_{t,j} - \gamma_{t+1}\pi_j$, such that, the chance to visit these regions will increase in the next iteration.

Although SAMC has been quite effective in sample space exploration, the convergence of θ_t is usually slow. Because, SAMC is run in a single chain. At each iteration, there is one and only one component of e_t is non-zero, and the information gained for θ_t is minimal and thus the adjustment process is slow. As a result, a large variation of θ_t can be observed even after long iterations, especially when the number of subregions is large.

Inspired by the successes of population-based MCMC algorithms, e.g., adaptive direction sampling [3], conjugate gradient Monte Carlo [4], parallel tempering [5,6], and evolutionary Monte Carlo [7,8], we propose a population SAMC (Pop-SAMC) algorithm to accelerate the convergence of SAMC. The new algorithm works on a population of SAMC chains. The benefits are two-fold. Firstly, it provides a more efficient self-adjusting mechanism. Intuitively, when we have a population of SAMC chains running in parallel, the information gained for θ_t at each iteration is increased, which leads to a more accurate adjustment of θ_t. Consequently, this improves the convergence of θ_t. Secondly, running a population of chains in parallel enables the incorporation of crossover operators from the genetic algorithm [9] into simulations. With this operator, the distributed information across a population can be shared among chains/population, and this improve the efficiency of the new algorithm.

In this paper, we illustrate the use of Pop-SAMC for Bayesian model selection problems using a change-point identification example and a gene selection example. The numerical results show that the new method performs significantly better than both the single chain SAMC and RJMCMC, in estimating probabilities of competing models. A rigorous proof for the convergence of Pop-SAMC is provided in Appendix.

The remainder of this paper is organized as follows. In Section 2, we describe the Pop-SAMC algorithm and study its convergence theory. In Section 3, we illustrate Pop-SAMC using a multimodal example. In Section 4, we show the superiority of Pop-SAMC for Bayesian model selection problems by studying a change-point identification example and a gene selection example, along with comparisons with SAMC and RJMCMC. In Section 5, we conclude this paper with a brief discussion.

Population SAMC Algorithm

Population SAMC

Consider the distribution defined in (1). Suppose the sample space χ has been partitioned into disjoint subregions, denoted by $E_1,\ldots,E_k$, and the same gain factor sequence $\{\gamma_t\}$ as defined in (3) and (4) for the single chain SAMC, will be used for Pop-SAMC.

Pop-SAMC works on a population of SAMC chains in parallel. At each iteration, a set of samples, called a population, are generated. Let $x^t = (x_1^t,\ldots,x_N^t)$ represent the population generated at iteration t, and N is the population size. One iteration of Pop-SAMC algorithm consists of the following two steps:

The Pop-SAMC algorithm

MH sampling: For each chain, simulate a sample x_i^t, for i=1,…,N, by a single MH update with the invariant distribution as defined in (5). A new population of samples X^t will be obtained.

Weight updating: Set $\theta^* = \theta_t + \gamma_{t+1}(\hat{\rho}_t - \pi)$ (7)

where $\hat{\rho}_t = (\sum_{i=1}^N I(x_i^t \in E_1)/N,\ldots,\sum_{i=1}^N I(x_i^t \in E_k)/N)'$. If $\theta^* \in \Theta$, set $\theta_{t+1} = \theta^*$; otherwise, set $\theta_{t+1} = \theta^* + c^*$, where $C^* = (c^*,\ldots,c^*)$ can be an arbitrary vector which satisfies the condition $\theta^* + C^* \in \Theta$.

Pop-SAMC is a generalized version of SAMC, with multiple independent samples (conditional on the current population) being generated at each iteration. This enables a frequency estimator for . Compared with the indicator vector et used in the single chain SAMC, $\hat{P}_t$ provides a more accurate estimator of P_t. This is the key reason why Pop-SAMC can outperform SAMC in terms of convergence of θ_t.

Convergence

Regarding the convergence of the algorithm, we note that for the empty subregions, the corresponding components of θ_t will trivially converge to -∞ as t →∞. Therefore, without loss of generality, we show in Appendix only the convergence of the algorithm for the case that all subregions are non-empty. Extending the proof to the general case is trivial, since replacing (7) by (8) (given below) will not change the process of pop-SAMC simulation.

$$\theta' = \theta_t + \gamma_t(\hat{p}_{t+1} - \pi - \nu), \quad (8)$$

where $V = (V,\ldots,V)'$ is an m-vector of V, and $\nu = \sum_{j\in\{i:E_i=\varnothing\}} \pi_j/(k-k_0)$ and k_0 is the number of empty subregions.

In our proof, we assume that Θ is a compact set; that is, there exists a constant vector c_t for each t such that $C_t + \theta_t \in \Theta$. This assumption is made only for the reason of mathematical simplicity. Extension of our results to the case that $\Theta = !^m$ is trivial with the technique of varying truncations studied in [10,11]. Interested readers can refer to [12] for the details, where the convergence of SAMC is studied with $\Theta = !^m$. In the simulations of this paper, we set $\Theta = [-10^{100}, 10^{100}]^m$, as a practical matter, this is equivalent to setting $\Theta = !^m$.

Under the above assumptions, we have the following theorem concerning the convergence of the Pop-SAMC algorithm, whose proof can be found in Appendix.

Theorem Let $E_1,\ldots,E_m$ be a partition of a compact sample space χ and $\psi(x)$ be a non-negative function defined on χ with $0 < \int_{E_i}\psi(x)dx < \infty$ for all E_i's. Let $\pi = (\pi,\ldots,\pi_m)$ be an m-vector with $0 < \pi_i < 1$ and $\sum_{i=1}^m \pi_i = 1$. Let $\{\gamma_t\}$ be a non-increasing, positive sequence satisfying (3). If Θ is compact and the drift condition [condition (A2) given in Appendix] is satisfied, then, as t→∞, we have almost surely,

$$\theta_{ti} \to \begin{cases} C + \log(\int_{E_i}\psi(x)dx) - \log(\pi_i + \nu), & \text{if } E_i \neq \varnothing, \\ -\infty, & \text{if } E_i = \varnothing. \end{cases} \quad (9)$$

where C is an unknown constant, $\nu = \sum_{j\in\{i:E_i=\varnothing\}} \pi_j/(k-k_0)$, and k_0 is the number of empty subregions.

The constant C can be determined by imposing a constraint, e.g., $\sum_{i=1}^{m} e^{\theta_{ti}}$ is equal to a known number.

The drift condition assumption is classical, which implies the existence of the stationary distribution $f_\theta(x)$ for any $\theta \in \Theta$. To have the drift condition satisfied, we assume that χ is compact and f(x) is bounded away from 0 and ∞ on χ. This assumption is true for many Bayesian model selection problems, for example, the Bayesian change-point identification and the Bayesian regression variable selection problems considered in this paper. For both problems, after integrating out the model parameters, the sample space is reduced to a finite set of models. For continuum systems, one may restrict χ to the region $\{x : \psi(x) \geq \psi_{\min}\}$, where $\psi_{\min}$ is sufficiently small such that the region $\{x : \psi(x) < \psi_{\min}\}$ is not of interest. Otherwise, one may put conditions on the tail of f(x) as prescribed by [11]. For the proposal distribution used in the MH sampling, we assume it to satisfy the local positive condition: There exists $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ such that $q(x, y) \geq \varepsilon_2$ if $|x - y| \leq \varepsilon_1$. This condition is quite standard and has been widely used in the study of MCMC convergence, [13].

Crossover

Another attractive feature of Pop-SAMC is that, with parallel independent running chains, information can be exchanged or shared between different chains to further improve the algorithm's efficiency. Borrowing information from different chains can be done through crossover operators originated in the genetic algorithm [9]. One pioneer work in this direction is the evolutionary Monte Carlo algorithm (EMC) [7,8]. Motivated by successes of the EMC, we incorporate crossover into Pop-SAMC and below we only discuss the simplest case for illustration purpose.

Suppose we have a population $X = (X_1, \ldots, XN)$ at iteration t, where $x_i = (a_i^1, \ldots, a_i^d)$ is a d-dimensional vector, the so-called an individual or chromosome. Let p_c denote the crossover rate, and $N_c = [N \times p_c]_e$ is the number of the chromosome in the current population that will be crossovered, where $[z]_e$ denotes the maximum even number smaller than z. For each iteration, the crossover operator works as follows:

Selection: Random select Nc chromosomes from the current population X, and randomly allocate them to form $N_c/2$ pairs.

Crossover: For each of the pairs, (x_i, x_j) $(i \neq j)$, an integer crossover point C is first decided by drawing uniformly from {1,…d}, then two new chromosomes (y_i, y_j) are obtained by swapping the components of the two parental chromosomes to the right of the crossover point. As shown below.

$$x_i = (a_i^1, \ldots, a_i^d) \qquad y_i = (a_i^1, \ldots, a_i^c, a_j^{c+1}, \ldots, a_j^d)$$

$$\Rightarrow$$

$$x_j = (a_j^1, \ldots, a_j^d) \qquad y_j = (a_j^1, \ldots, a_j^c, a_i^{c+1}, \ldots, a_i^d)$$

Following MH rules, the new chromosomes are accepted into the new population y with probability $\min(1, r_c)$, and

$$r_c = \frac{f_{\theta_t}(y_i) f_{\theta_t}(y_j)}{f_{\theta_t}(x_i) f_{\theta_t}(x_j)} \times \frac{T((x_i, x_j) \mid (y_i, y_j))}{T((y_i, y_j) \mid (x_i, x_j))} \tag{10}$$

where $T((y_i, y_j) \mid (x_i, x_j)) = P((x_i, x_j) \mid \boldsymbol{x}) P((y_i, y_j) \mid (x_i, x_j))$. $P((x_i, x_j) \mid \boldsymbol{x})$ is the select probability of (x_i, x_j) from the population x and $P((y_i, y_j) \mid (x_i, x_j))$ is the generating probability of (y_i, y_j) from the parental chromosomes (x_i, x_j). Since our parental chromosomes are chosen randomly from the population and by the symmetric properties of the crossover operator, it is easy to show that $T((y_i, y_j) \mid (x_i, x_j)) = T((x_i, x_j) \mid (y_i, y_j))$. Thus, equation (10) will be reduced to the likelihood ratio between the new chromosomes and the old ones:

$$r_c = \frac{f_{\theta_t}(y_i) f_{\theta_t}(y_j)}{f_{\theta_t}(x_i) f_{\theta_t}(x_j)} \tag{11}$$

In the Pop-SAMC, the crossover operation can be included in the MH Sampling step, N_c chromosomes are updated using the crossover operator, and $(N\text{-}N_c)$ chromosomes are updated with a single MH step respectively according to the invariant distribution as defined in (5).

The rationale behind the effectiveness of crossover operators can be explained as follows. At each iteration, some samples obtained in one chain may be better than others in terms of likelihood values. If this chain happens to be selected for the crossover operation, by exchanging parts of its chromosome with other individuals, the overall quality of the whole population could be potentially improved.

An Illustrative Example

To illustrate the performance of Pop-SAMC in estimation of the sample space partitioning weight, we study a multimodal example [8], whose density function is given by

$$f(x) = \frac{1}{2\pi\sigma^2} \sum_{i=1}^{20} \alpha_i \exp\left\{-\frac{1}{2\sigma^2}(x - \mu_i)(x - \mu_i)\right\} \tag{12}$$

where each component has an equal variance $\sigma^2 = 0.02$ and is assigned an equal weight $\alpha_1 = \ldots = \alpha_{20} = 0.05$. See [8] for the values of the mean vectors.

It is shown that some components are far from others, more than 30 times of the standard deviation in distance. This puts a great challenge on the testing algorithm.

Let $\mathcal{X} = [-10^{100}, 10^{100}]^2$, and let it be partitioned according to $U(x) = -\log\{f(x)\}$, (In terms of physics, $U(X)$ is called the energy function of the distribution), with an equal energy bandwidth $\Delta u = 0.5$ into the following subregions: $E_1 = \{x : u(x) < 0\}, E_2 = \{x : 0 \leq u(x) < 0.5\}, \ldots, E_{50} = \{x : u(x) > 24.0\}$ and the desired sampling distribution to be uniform $\pi_1 = \ldots = \pi_{50} = \frac{1}{50}$. Both Pop-SAMC and SAMC provide a self-adjusting mechanism for learning the partition weights $\int_{E_i} f(x)dx / \pi_i$, for $i = 1, \ldots, 50$, in an online manner. With the uniform desired sampling distribution, the partition weight reduce to the probability that a sample is drawn from each subregion i, i.e. $P(E_i) = \int_{E_i} f(x)dx$. The true value of $P(E_i)$ is calculated with a total of 20×10^8 samples drawn equally from each of the twenty components of f(x). In order to have a fair comparison, we run both SAMC and Pop-SAMC with the same number of energy evaluations, using the same proposal distribution and the same gain factor sequences.

Pop-SAMC was run 100 times independently with the setting: $T_0 = 50, N = 10, K = 50$, and the number of iterations $= 10^5$. SAMC was applied to this example 100 times independently with the same setting except for the following parameters: $T_0 = 100$ and the number of iterations $= 10^6$. The computational results along with the true value of $P(E_i)$'s for $i = 2, \ldots, 9$, are summarized in (Table 1), the results for other subregions with zero or tiny probability are not listed. The results show

Estimates	True Prob. (%)	Pop-SAMC	SAMC
$P(E_2)$	23.87	23.85(0.05)	23.65(0.85)
$P(E_3)$	30.27	30.25(0.06)	30.31(0.92)
$P(E_4)$	18.56	18.59(0.04)	18.13(0.46)
$P(E_5)$	11.24	11.21(0.02)	11.30(0.47)
$P(E_6)$	6.63	6.64(0.02)	6.27(0.12)
$P(E_7)$	3.84	3.85(0.01)	3.63(0.07)
$P(E_8)$	2.26	2.26(0.01)	2.15(0.04)
$P(E_9)$	1.34	1.34(0.00)	1.27(0.02)
CPU(s)	---	1.81	2.36

Table 1: Comparison of SAMC and Pop-SAMC for the multimodal example. The number in the parentheses is the standard deviation of the corresponding estimate. CPU: the CPU time (in seconds) cost by a single run of the corresponding algorithm on a Intel Core 2 Duo 3.0 GHz computer.

that Pop-SAMC has made a significant improvement in accuracy over SAMC in terms of standard deviations of the estimates of $P(E_i)$'s. On average, the standard error of the Pop-SAMC estimates is only about 1/10 of that of the SAMC estimates. These results are achieved under the same number of energy evaluations for each of the algorithm, which made the comparison fair. In the table, we also reported the CPU times cost by a single run of each method to further clarify the fairness of our comparison. Pop-SAMC even cost less CPU time than SAMC in this comparison. In summary, Pop-SAMC can converge much faster than SAMC for estimation of the weight of subregions.

Bayesian Model Selection Problems

A Bayesian approach to model selection problems proceeds as follows. Suppose that we have a posterior distribution denoted by $P(m \mid \boldsymbol{y}) \propto P(m) f(\boldsymbol{y} \mid m)$, where y denotes the data, m is the model index, which belongs to a set of competing models, $m \in M$, and $P(m)$ is the prior probability of model m, $f(y/m)$ is the marginal likelihood, which is obtained by integrating out the model parameters. Different methods have been developed to estimate the posterior probability of all potential models. The criteria to judge each method is based on the accuracy of their estimation.

SAMC has been compared with RJMCMC in Bayesian model selection problem [2]. The results show that SAMC outperforms RJMCMC when the model space is complex. However, when the model space is simple, e.g., it only contains several models with comparable probability, SAMC may not be better than RJMCMC. On the other hand, compared with SAMC, Pop-SAMC has shown its superiority in estimating sample space partition weight. For model selection problems, the sample space is usually partitioned according to the model index by assuming that the MH chain can mix reasonably well in the sample space of each model. The weight of each partition is thus proportional to the posterior probability $P(m/y)$ of each potential model. Accordingly, Pop-SAMC is expected to be more efficient in estimating the model probability than SAMC. In this section, we show the superiority of Pop-SAMC in Bayesian model selection problems by studying two typical examples. The results show that Pop-SAMC can make a significant improvement over SAMC and it can also work better than RJMCMC even when the model space is simple.

A change-point identification example

The change-point identification problem can be described as follows. Suppose we have a sequence of independent observations $y = (y_1, y_2, \ldots, y_n)'$ and they can be partitioned into blocks, such that the sequence follows the same distribution within blocks. Our goal is to identify the unknown number and the locations of the boundary, called change-point, between blocks. For simplicity, we assume that the observations within each block are drawn independently from a normal distribution $N(\mu_b, \sigma_b^2)$, where b is the index of blocks. After a change-point, both the mean and variance may shift.

In the literature, this problem has been studied by several authors using simulation-based methods, e.g., the Gibbs sampler [14], reversible jump MCMC [1], jump diffusion [15], and evolutionary Monte Carlo [7]. In this paper, we follow [7]'s approach, a latent vecotor is introduced to indicate the change-point position. Let $Z = (Z_1, \ldots, Z_{n-1})$ be a latent binary vector associated with the observations index except for the last one, indicating the potential change-point, where $Z_i = 1$ indicates a change-point, and 0 otherwise. Let Z(k) correspond to a model with k change-point, with the unknown change-points being denoted by $C_1, \ldots, C_k$. For convenience, we let $C_0 = 0$ and $C_{k+1} = n$ $c_0 < c_1 < c_2 < \ldots < c_k < c_{k+1}$ and they follow the order .

Under the above setting, we have $y_i \sim N(\mu_b, \sigma_b^2), \quad c_{b-1} < i \le c_b$ (13) for b =1,2,…,k+1 and i=1,…,n. For model $Z^{(k)}$, the parameter vector is $\theta^{(k)} = (z^{(k)}, \mu_1, \sigma_1^2, \ldots, \mu_{k+1}, \sigma_{k+1}^2)$. Let χ_k denote the model space with k change-points, $Z^{(k)} \in \chi_k$ and $\mathcal{X} = \bigcup_{k=0}^{n-1} \mathcal{X}_k$. The log-likelihood function of model $\theta(k)$ is then

$$L(y \mid \theta^{(k)}) = -\sum_{i=1}^{k+1} \left\{ \frac{c_i - c_{i-1}}{2} \log \sigma_i^2 + \frac{1}{2\sigma_i^2} \sum_{j=c_{i-1}+1}^{c_i} (y_i - \mu_i)^2 \right\} \quad (14)$$

To conduct a Bayesian analysis for the model, we specify the following prior distribution for the model parameters:

$$\sigma_i^2 \sim IG(\alpha, \beta), P(\mu_i) \propto 1 \quad (15)$$

where $IG(\cdot, \cdot)$ denotes an inverse Gamma distribution with hyperparameters α,β, and an improper uniform prior is put on each μ_i. In addition, we assume that the latent vector $Z^{(k)}$ follows a truncated Poisson distribution,

$$P(z^{(k)}) \propto \frac{\lambda^k}{\sum_{j=0}^{n-1} \frac{\lambda^j}{j!}} \frac{(n-1-k)!}{(n-1)!}, \qquad k = 0, 1, \ldots, n-1. \quad (16)$$

Where λ is a hyperparameter; n-1 is the largest number of change-points allowed by this model. Conditioning on the number of change-points k, we put an equal prior probability on all possible configurations of $Z^{(k)}$. By assuming that all the priors are independent, the log-prior density is

$$P(\theta^{(k)}) = a_k - \sum_{i=1}^{k+1} \left\{ (\alpha + 1) \log \sigma_i^2 + \frac{\beta}{\sigma_i^2} \right\} \quad (17)$$

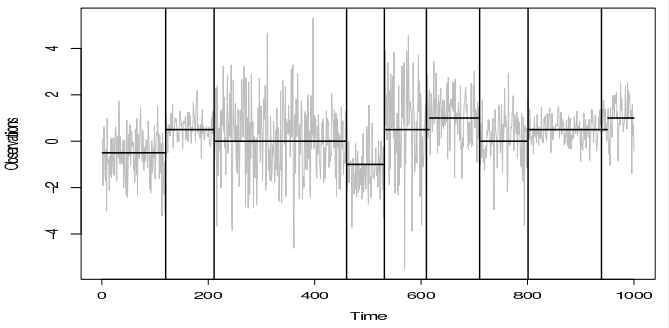

Figure 1: A comparison of the true change-point position (horizontal line) and the MAP estimates (vertical line).

where $a_k = (k+1)\{\alpha \log\beta - \log\Gamma(\alpha)\} + \log(n-1-k)! + k\log\lambda$. Combining the likelihood (14) and prior distributions (17), integrating out μ_i and σ_i^2 for $i = 1,\ldots,k+1$ and taking the logarithm, we get the following log-posterior density function

$$\log P(z^{(k)} \mid y) = a_k + \frac{k+1}{2}\log 2\pi - \sum_{i=1}^{k+1}\left\{\frac{1}{2}\log(c_i - c_{i-1}) - \log\Gamma\left(\frac{c_i - c_{i-1} - 1}{2} + \alpha\right) + \left(\frac{c_i - c_{i-1} - 1}{2} + \alpha\right)\log\left[\beta + \frac{1}{2}\sum_{j=c_{i-1}+1}^{c_i} y_j^2 - \frac{\left(\sum_{j=c_{i-1}+1}^{c_i} y_j\right)^2}{2(c_i - c_{i-1})}\right]\right\} \quad (18)$$

Samples generated from the above posterior distribution can be used to estimate P(χ_k/y). For Pop-SAMC, if we let $E_k = \mathcal{X}_k$ and $\psi(\cdot) \propto P(z^{(k)} \mid y)$, it follows from (9) that $\hat{w}_i^{(t)} / \hat{w}_j^{(t)} = e^{\theta_{it} - \theta_{jt}}$ forms a consistent estimator for the posterior probability ratio $P(\mathcal{X}_i \mid \boldsymbol{y}) / P(\mathcal{X}_j \mid \boldsymbol{y})$. Without loss of generality, we restrict our consideration to the models with $k_{\min} \le i, j \le k_{\max}$, where k_{min} and k_{max} can be determined easily with a short pilot run of the above algorithm, the probability of those models outside this range is zero.

For the change-point identification problem, the details of the sampling step of Pop-SAMC are designed similarly to those described in [16], except for the weight updating step, which follows (9).

In this example, the simulated dataset consists of 1000 observations with $y_1,\ldots,y_{120} \sim N(-0.5,1)$, $y_{121},\ldots,y_{210} \sim N(-0.5,0.5)$, $y_{211},\ldots,y_{460} \sim N(0,1.5)$, $y_{461},\ldots,y_{530} \sim N(-1,1)$, $y_{531},\ldots,y_{615} \sim N(0.5,2)$, $y_{616},\ldots,y_{710} \sim N(1,1)$, $y_{711},\ldots,y_{800} \sim N(0,1)$, $y_{801},\ldots,y_{950} \sim N(0.5,0.5)$, and $y_{951},\ldots,y_{1000} \sim N(1,1)$. The time plot is shown in (Figure 1). For this example, we set the hyperparameters $\alpha = \beta = 0.5$, which forms a vague prior for σ_i^2; and set λ=1. After a short pilot run, we set $k_{min} = 7$ and k_{max} =14

Pop-SAMC was run for this example 50 times independently with the following setting:

N=20, T_0=10, Iterations = 5×10^4 and $\pi_1 = \ldots = \pi_8 = \frac{1}{8}$. The results are summarized in (Figure 1) and (Table 2). Figure 1 shows the comparison between the eight true change-point pattern with its MAP (maximum a posteriori) estimate, which are (120, 210, 460, 530, 615, 710, 800, 950) and (120, 211, 460, 531, 610, 710, 801, 939) respectively. The two patterns match very well except for the last point. A detailed exploration of the simulated dataset gives a strong support to the MAP estimate. The last ten observations of the second to the last block have a larger mean value than the expected and thus, they have been grouped into the last block. The MAP estimates also achieves larger log-posterior probability than that of the true pattern, which is 5305.57>5300.24.

For comparison, SAMC and RJMCMC were also applied to this example. Each algorithm was run 50 times independently. The results are summarized in (Table 2). SAMC employs the same setting as Pop-SAMC except for two parameters, T_o=100 , Iterations=10^6 . RJMCMC employs the same transition proposals as those used by Pop-SAMC and SAMC and performs 10^6 iterations in each run. Under these settings, for a single run, each of the three algorithms performs exactly the same number of energy evaluations with the same transition proposals. Therefore, the comparisons made in (Table 2) are fair to each of the algorithm. The CPU time cost by each run is about the same for all methods.

The comparison shows that Pop-SAMC works best among these three methods, with the smallest standard error achieved in estimating the posterior probability P(χ_k/y). As expected, RJMCMC works better than SAMC in this example. Because for this example, the model space is quite simple, containing only one mode with comparable probabilities. As previously mentioned, under such a situation, SAMC may not be better than RJMCMC. However, Pop-SAMC does. Although, Pop-SAMC is essentially an important sampling method as SAMC, its improved self-adjusting mechanism makes it much more efficient than SAMC. Amazingly, this improvement in its ability to learn from past samples enables Pop-SAMC to even conquer RJMCMC for those problems in which RJMCMC succeeds.

It is worth pointing out that, both Pop-SAMC and SAMC beat RJMCMC in the low probability model spaces, e.g. k =7,13,14, even though SAMC is worse than RJMCMC overall. The reason is the following. RJMCMC does not have self-adjusting ability: it samples each model in a frequency proportional to its probability. In contrast, due to their self-adjusting mechanism, Pop-SAMC and SAMC sample equally from each model space, they work well for the low probability model space as well as for the high probability part.

Finally, in order to further increase Pop-SAMC's efficiency and fully use the information among the population, we incorporate crossover operator into the computation with 10% crossover rate and keep all other settings intact. The algorithm was run 50 times independently, and the results were also included in (Table 2) for comparison. Unsurprisingly, Pop-SAMC works more efficiently with the crossover operator.

A Large Regression Model Selection Example

To have a further assessment of the performance of the Pop-SAMC in Bayesian model selection problems, we consider a linear regression variable selection example, in which the number of observations n is much less than the number of potential predictors p.

The linear regression model with a fixed number of potential

	Pop-SAMC 10%Cr		Pop-SAMC		SAMC		RJMCMC	
k	prob(%)	SD	prob(%)	SD	prob(%)	SD	porb(%)	SD
7	0.1029	0.0014	0.1009	0.0018	0.0949	0.0026	0.0998	0.0052
8	55.5077	0.2272	55.6082	0.2698	54.5699	0.6833	55.0832	0.3261
9	33.3677	0.1364	33.2264	0.1693	33.5970	0.4432	33.5365	0.1794
10	9.2642	0.0873	9.3098	0.1010	9.8146	0.2910	9.4942	0.1548
11	1.5646	0.0253	1.5633	0.0233	1.7117	0.0778	1.5884	0.0547
12	0.1767	0.0037	0.1756	0.0031	0.1943	0.0113	0.1813	0.0108
13	0.0150	0.0004	0.0149	0.0003	0.0165	0.0011	0.0153	0.0013
14	0.0010	0.0000	0.0010	0.0000	0.0012	0.0001	0.0012	0.0002
CPU(s)	16.1		16.2		16.2		15.2	

Table 2: Comparison of the estimated posterior distribution for the change-point identification example. The number in the parentheses is the estimates of standard deviation (SD). CPU: the CPU time (in seconds) cost by a single run of the corresponding algorithm on a Intel Core 2 Duo 3.0 GHz computer.

	Pop-SAMC 10%Cr		Pop-SAMC		SAMC		RJMCMC	
k	prob(%)	SD	prob(%)	SD	prob(%)	SD	porb(%)	SD
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2125	0.3689
11	0.0000	0.0000	0.0000	0.0000	0.0012	0.0035	0.1345	0.2223
12	0.0000	0.0000	0.0000	0.0000	0.0052	0.0135	0.1515	0.1636
13	0.0001	0.0002	0.0001	0.0002	2.0299	6.5153	45.6895	8.4369
14	0.0001	0.0002	0.0001	0.0001	1.4000	4.4203	38.0850	6.8908
15	0.0000	0.0000	0.0000	0.0001	0.2382	0.7520	7.3470	1.2674
16	0.0000	0.0000	0.0000	0.0000	0.0245	0.0712	1.0135	0.2736
17	0.0000	0.0000	0.0000	0.0001	0.0082	0.0128	1.5905	5.0314
18	0.0000	0.0000	0.0000	0.0001	0.0135	0.0260	1.1135	1.5058
19	0.0000	0.0000	0.0000	0.0002	0.0059	0.0099	0.9490	1.0440
20	0.0000	0.0000	0.0000	0.0001	0.0021	0.0040	0.3740	0.5632
21	0.0000	0.0000	0.0000	0.0000	0.0005	0.0011	0.0930	0.1583
22	0.0000	0.0000	0.0000	0.0000	0.0001	0.0003	0.0190	0.0361
23	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0035	0.0114
24	0.0001	0.0002	0.0001	0.0003	0.0080	0.0177	0.0005	0.0022
25	0.0000	0.0001	0.0001	0.0001	0.0029	0.0068	0.0005	0.0022
26	0.0001	0.0001	0.0001	0.0000	0.0008	0.0016	0.0000	0.0000
27	83.5210	1.0919	84.1339	1.4828	81.0277	10.0109	2.6935	12.0457
28	14.9215	0.9560	14.3358	1.2640	13.7389	2.2726	0.4760	2.1287
29	1.4415	0.1639	1.4189	0.2248	1.3769	0.2886	0.0460	0.2057
30	0.1080	0.0211	0.1037	0.0188	0.1077	0.0272	0.0040	0.0179
31	0.0071	0.0018	0.0066	0.0014	0.0071	0.0022	0.0000	0.0000
32	0.0004	0.0002	0.0004	0.0001	0.0004	0.0002	0.0000	0.0000

Table 3: Comparison of the estimated posterior distribution $P(x_k \mid \boldsymbol{y})$ for the simulated large P linear regression example. The number in the parentheses is the estimates of standard deviation (SD).

predictors $\{x_1,x_2,\ldots,x_p\}$ takes the form

$$y = X\beta + \varepsilon, \qquad \varepsilon \sim N_n(0, I_n / \tau) \tag{19}$$

where $y = (y_1, y_2, \ldots, y_n)'$ is the response vector, $X = [1, x_1, \ldots, x_p]$ is an $n \times (p+1)$ design matrix, and $\beta = (\beta_0, \beta_1, \ldots, \beta_p)'$ is a $(p+1)$ -vector of regression coefficients. The problem of interest is to find a subset model M_k of the form

$$y = X_k\beta_k + \varepsilon, \qquad \varepsilon \sim N_n(0, I_n / \tau) \tag{20}$$

which is best under some criterion, where $0 \le k \le p, X_k = [1, x_1^*, \ldots, x_k^*], x_1^*, \ldots, x_k^*$, are the selected predictors and $\beta_k = (\beta_0^*, \beta_1^*, \ldots, \beta_k^*)'$ is the vector of regression coefficients of the subset model. For model M_k, the likelihood function is

$$L_k(y \mid X_k, \beta_k, \tau, M_k) = \left(\frac{\tau}{2\pi}\right)^{n/2} \exp\left\{-\frac{\tau}{2}(y - X_k\beta_k)'(y - X_k\beta_k)\right\}. \tag{21}$$

The prior distributions for each parameter are assigned as follows. We first assume τ and β_k are subject to the *g* priors [17],

$$P(\tau) \propto \frac{1}{\tau}, \qquad \beta_k \mid \tau, M_k \sim N\left(0, \frac{g}{\tau}(X_k'X_k)^{-1}\right), \tag{22}$$

where *g* is a hyperparameter. We further assume that all the p predictors are linearly independent, and each has the same prior probability q to be included in the model. Therefore, the prior probability imposed on the mode M_k is

$$P(M_k) = q^k(1-q)^{p-k}, \tag{23}$$

with q being subject to the uniform distribution *Unif* [0,1].

Collecting the likelihood and prior distributions, we get the posterior distribution,

$$P(M_k, \tau, \beta_k, q \mid y) \propto L_k(y \mid X_k, \beta_k, \tau, M_k)P(\tau)P(\beta_k \mid X_k, \tau, g)P(M_k \mid q)P(q) \tag{24}$$

Integrating out τ, βk and q from (24) and taking the logarithm, we get the log-posterior of model M_k (up to an additive constant),

$$\log P(M_k \mid y) = \log\Gamma(k+1) + \log\Gamma(p-k+1) - \frac{k}{2}\log(1+g) - \frac{n}{2}\log\left[y'y - \frac{g}{1+g}y'X_k(X_k'X_k)^{-1}X_k'y\right] \tag{25}$$

where g is specified by the user, which reflects their prior knowledge on the model space. Typically, large g concentrates the prior on parsimonious models with a few large coefficients, and small g tends to concentrate the prior on saturated models with small coefficients [18]. The evaluation of the posterior distribution involves inverting a matrix, which can be calculated in a recursive manner using the matrix inversion in block form, and this will save the computation cost tremendously.

Simulation study

The small *n* large *p* example is modified from some examples studied in [19,20]. The dataset is generated as follows. Let $z_i \sim N_{150}(0, I)$ for i = 1,…,600 and define

$$\begin{aligned}
x_i &= z_i, \quad i = 1,\ldots,30;\\
x_i &= z_i + 0.3z_{i-30} + 0.5z_{i-29} - 0.7z_{i-28} + 0.9z_{i-27} + 1.1z_{i-26} + 0.2z_{i+80}\\
&\quad -0.4z_{i+180} + 0.6z_{i+280} - 0.8z_{i+380} + z_{i+480}, \quad i = 31,\ldots,40;\\
x_i &= z_i + 0.3z_{i-30} + 0.5z_{i-29} + 0.7z_{i-28} - 0.9z_{i-27} + 1.1z_{i-26} + 0.2z_{i+80}\\
&\quad +0.4z_{i+180} - 0.6z_{i+280} + 0.8z_{i+380} - z_{i+480}, \quad i = 41,\ldots,50;\\
x_i &= z_i + 0.3z_{i-30} - 0.5z_{i-29} + 0.7z_{i-28} + 0.9z_{i-27} + 1.1z_{i-26} - 0.2z_{i+80}\\
&\quad +0.4z_{i+180} + 0.6z_{i+280} + 0.8z_{i+380} + z_{i+480}, \quad i = 51,\ldots,60;\\
x_i &= z_i, \quad i = 61,\ldots,600;
\end{aligned}$$

The response variable is defined as

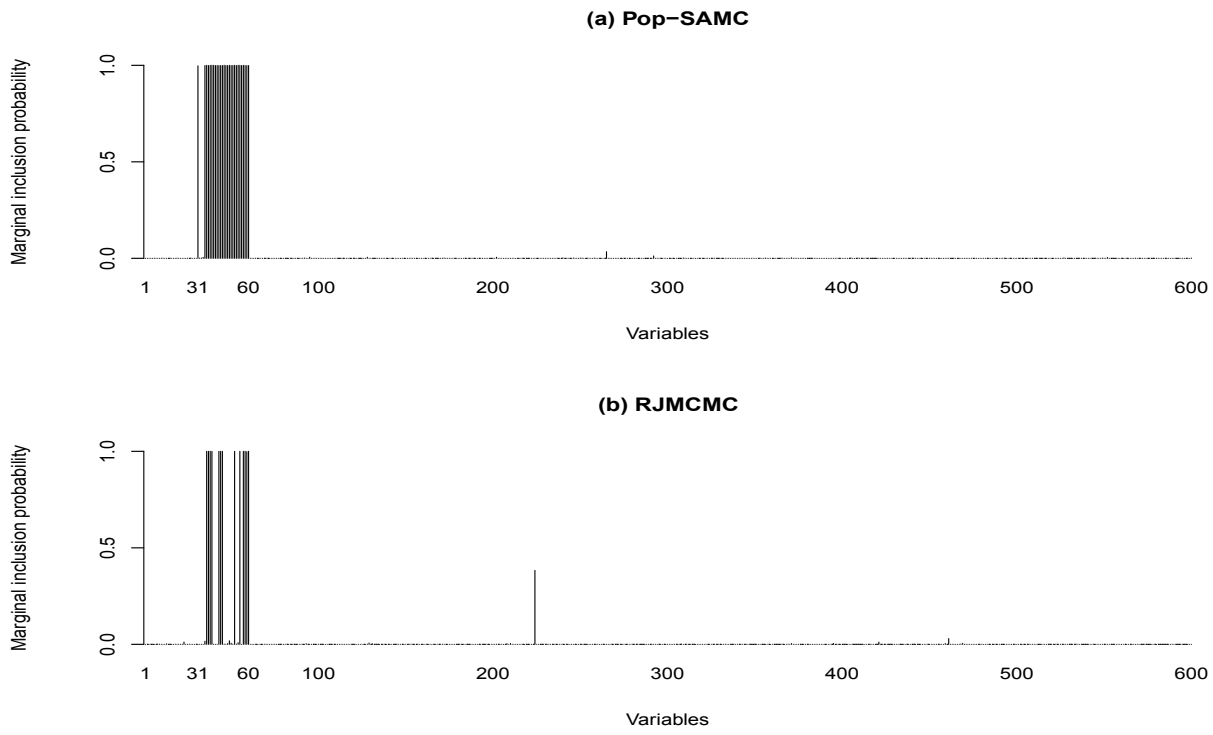

Figure 2: Estimates of marginal inclusion probabilities produced by (a) Pop-SAMC and (b) RJMCMC in a single run.

$$y = 1 + \sum_{i=31}^{60} x_i + \varepsilon, \tag{27}$$

where $\varepsilon \sim N_{150}(0, 4I)$, and is independent of other predictor variables.

For this example, we set the hyperparameter $g = \max(n, p^2)$, the so called benchmark prior recommended by [19]. Given the full posterior distribution, in applying Pop-SAMC to this example, we follow the same fashion as in the change-point identification example. We first partition the sample space according to the model index k. Let $E_k = \mathcal{X}_k$, the model space with k selected variables, $M_k \in \mathcal{X}_k$, and $\psi(\cdot) \propto P(M_k \mid y, \eta_\tau)$. It follows from (9) that $\hat{w}_i^{(t)} / \hat{w}_j^{(t)} = e^{\theta_{ti} - \theta_{tj}}$ forms a consistent estimator for the posterior probability ratio $P(\mathcal{X}_i \mid y) / P(\mathcal{X}_j \mid y)$. We restrict the model space to be $k_{\min} \le i, j \le k_{\max}$. After a pilot run, we set k_{min} = 10 and k_{max} =40.

Again, we applied four algorithms to this example and compare their efficiency. Each algorithm was run 20 times independently. Firstly, Pop-SAMC was run for this example with the following setting: N=20, T_0= 400, Iterations = 3× 10^5 and Then, we include crossover operator into Pop-SAMC. The modified algorithm was run for this example with 10% crossover rate while keeping all other settings intact. Finally SAMC and RJMCMC were applied to this example respectively. SAMC employs the same setting as Pop-SAMC except for one parameter, Iterations = 6× 10^6. RJMCMC also performs 6× 10^6 iterations in each run, with the first 50000 iterations being discarded for the burnin process. For all the four algorithms, the same transition proposal is used, and for a single run, each of them performs exactly the same number of energy evaluations. Therefore, the comparisons are fair to each of the algorithms. The computational results are summarized in (Table 3), those subregions with zero probability for all the algorithms are not listed.

The comparison shows that the order of the four algorithm in terms of efficiency for this example is Pop-SAMC with 10% crossover > Pop-SAMC > SAMC> RJMCMC. This order is consistent with that in the change-point identification example, except SAMC beats RJMCMC this time. In fact, RJMCMC failed in this example, it chose the model M_{13} instead of the true model M_{27}. This is because there are two modes in the model space, which are well separated. For RJMCMC, it does not have the self-adjust ability, which makes it easily get trapped into a local mode. However, all the other three algorithms have a self-adjusting mechanism, which enables them to get out of local trap and explore the whole sample space quickly. The efficiency improvement for Pop-SAMC based algorithms over SAMC is also significant, especially at the true mode and low probability model space, e.g. k = 11:26.

We further check the estimates of the marginal inclusion probabilities produced by Pop-SAMC and RJMCMC in a single run, which is shown in (Figure 2). From this plot, we may tell that all the 27 variables selected by Pop-SAMC are in the true variables rang from 31 to 60. Due to the correlation among the variable set, variable 32, 33 and 34 were not selected. On the other hand, the variables selected by RJMCMC are also belong to the true variable set, but it only found 13 out of the 27.

A Real data analysis

The dataset we studied here was generated by [21]. As described in [22], the experiment concerns the genetic basis for differences between two inbred mouse populations (B6 and BTBR). Based on their in-house selective phenotyping algorithm, 60 (B6×BTBR) $F_2 - ob / ob$ mice (29 males and 31 females) were selected. A total of 60 arrays were used

	Pop-SAMC		SAMC		RJMCMC	
k	prob(%)	SD	prob(%)	SD	porb(%)	SD
1	0.2792	0.1049	0.1991	0.0948	0.1810	0.0907
2	16.0970	1.9719	14.6317	4.8154	13.6790	2.5625
3	20.2100	3.2758	18.2059	4.2074	18.0265	3.3172
4	35.5714	4.0382	36.0330	8.9530	39.0120	5.0084
5	16.9414	1.1698	17.0727	2.1761	17.8005	1.7256
6	6.7641	0.8092	7.0985	1.9672	6.5390	0.9450
7	2.5384	0.4956	3.3053	1.9966	2.5420	1.4040
8	0.9915	0.2542	1.9386	2.0487	1.3045	2.6034
9	0.4478	0.1443	1.1381	1.4715	0.7115	2.1479
10	0.1591	0.0853	0.3770	0.4761	0.2045	0.7313

Table 4: Comparison of the estimated posterior distribution $P(\mathcal{X}_k \mid \boldsymbol{y})$ for the real large linear regression example. The number in the parentheses is the estimates of standard deviation (SD).

to monitor the expression levels of 22,690 genes. Some physiological phenotypes were also measured by quantitative real-time RT-PCR, e.g. the numbers of stearoyl-CoA desaturase 1 (SCD1). The raw data are available in GEO (http://www.ncbi.nlm.nih.gov/geo; accession number GSE3330).

We treat the phenotypic value (SCD1) as the dependent variable, and the expression levels of genes as predictors. The value of SCD1 was first adjusted to remove the possible gender effects, then its correlation to each gene is calculated. We ordered the genes according to the correlation from high to low, and took the first 1000 genes as the potential predictors.

Three algorithms were applied to this dataset, Pop-SAMC, SAMC and RJMCMC. Each algorithm was run for this example 20 times independently. Follow the procedure in the simulation study, the sample space was partitioned according to the model index, and after a few pilot runs, we restricted the model space to be $1 \le k \le 10$.In the pilot runs, we also found, with such a small number of observations, $n = 60$, setting $g = p^2$ made the penalty to complex models so strong such that the mode was pushed around 1. In order to consider more potential models, we relaxed the penalty in priors and set $g = p = 1000$. The parameters for each algorithm were set as follows. For Pop-SAMC, N=10, T_0=20 , Iterations = 6×10^4, and $\pi_1 = \ldots = \pi_{10} = \frac{1}{10}$; SAMC employs the same setting as Pop-SAMC except T_0=50 and Iteration = 6×10^5; RJMCMC also performs 6×10^5 iterations with the first 20000 iterations as warming. As in the simulation study, the same transition proposal is used for all the algorithms, and for a single run, each of them performs exactly the same number of energy evaluations. Therefore, the comparison is fair. The computational results are summarized in (Table 4).

The results is clear, Pop-SAMC works best among the three methods with the smallest standard error achieved in estimating the posterior probability of potential models. Since there is only one mode in the model space for this dataset, as an important sampling algorithm, SAMC can not beat RJMCMC overall, it only won the battle in the low probability model space. However, with the improved self-adjusting mechanism, Pop-SAMC has conquered RJMCMC in the whole model space.

Discussion

In this paper, we have proposed a Pop-SAMC algorithm, it works on a population of SAMC chains. Compared with single chain SAMC, the generalized algorithm can converge much faster. The superior efficiency of the new algorithm is demonstrated by Bayesian model selection problems. Our numerical results show that Pop-SAMC significantly outperforms both single chain SAMC and RJMCMC.

The success of the Pop-SAMC is based on two features. Firstly, the so-called population effect. Since it works with a population of SAMC chains, at each iteration, by integrating the information of all samples from each chain, it provides a more accurate estimation of P_t, therefore, the self-adjusting mechanism is more efficient. Secondly, running a population of chains in parallel enables incorporation of advanced operators, such as the crossover operator, snooker operator and gradient operator. With these population-based operators, the distributed information across a population at each iteration can be shared globally, which may further increase the algorithm's efficiency. In addition, for each chain, Pop-SAMC keeps the self-learning feature of SAMC, which operates based on past samples.

There remain much improvement space and a broader field of applications for the new algorithm. For instances, how to combine the multiple-try method with each chain of Pop-SAMC to increase the accept probability in dealing with high dimension space problems; are there any guidance to determine the population size and crossover rate instead of the trial and error scheme used in this paper? Regarding the application of the new algorithm, a promising research field is to design a Pop-SAMC based particle filter.

The population setting of this algorithm provides good basis to develop advanced particle filter; moreover, Pop-SAMC has kept the two attractive feature of SAMC (i) superiority in sample space exploration and (ii) ability to generate weight-bounded importance samples. By taking advantage of these features, the notorious weight degeneracy problem encountered by traditional particle filter has a very good chance to be overcome. The authors are actively working on this field now.

Acknowledgment

Liang's research was supported in part by the grant (DMS-1007457 and CMMI-0926803) made by the National Science Foundation and the award (KUS-C1-016-04) made by King Abdullah University of Science and Technology (KAUST). The authors thank the editor, the associate editor, and the referee for their comments which have led to significant improvement of this paper.

References

1. Green P (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 82: 711-732.
2. Liang F, Liu C, Carroll RJ (2007) Stochastic approximation in Monte Carlo computation. J Am Statist Ass 102: 305-320.
3. Gilks WR (1994) Adaptive direction sampling. The Statistician 43: 179-189.
4. Liu JS, Liang F, Wong WH (2000) The use of multiple-try method and local optimization in Metropolis sampling. J Am Statist Ass 95: 121-134.
5. Geyer CJ (1991) Markov chain Monte Carlo maximum likelihood. in Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface, (ed EM Keramigas), pp 153-163.
6. Hukushima K, Nemoto K (1996) Exchange Monte Carlo method and application to spin glass simulations. J Phys Soc Jpn 65: 1604-1608.
7. Liang F, Wong WH (2000) Evolutionary Monte Carlo sampling: applications to Cp model sampling and change-point problem. Statistica Sinica 10: 317-342.
8. Liang F, Wong WH (2001) Real-parameter evolutionary sampling with applications in Bayesian Mixture Models. J Am Statist Ass 96: 653-666.

9. Holland JH (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press Ann Arbor.

10. Chen HF (2002) Stochastic approximation and its applications, Boston: Kluwer Academic Publishers.

11. Andrieu C, Moulines Ě, Priouret P (2005) Stability of stochastic approximation under verifiable conditions. SIAM J Control and Optimization 44: 283-312.

12. Liang F, Liu C, Carroll, RJ (2010) Advanced Markov chain Monte Carlo: Learning from Past Samples. Wiley, ISBN: 978-0-470-74826-8.

13. Roberts GO, Tweedie RL (1996) Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. Biometrika 83: 95-110.

14. Barry D, Hartigan JA (1993) A Bayesian analysis for change point problems. J Am Statist Ass 88: 421.

15. Phillips DB, Smith AFM (1996) Bayesian model comparison via jump diffusions. in Markov Chain Monte Carlo in Practice (WR Gilks, S Richardson, DJ Spiegelhalter, eds.) 215-239 London: Chapman & Hall.

16. Liang F (2009) Improving SAMC using smoothing methods: theory and applications to Bayesian model selection problems. The Annals of Statistics 37: 2626-2654.

17. Zellner A (1986) On assessing prior distributions and Bayesian regression analysis with g-prior distributions. in Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti, eds. PK Goel, A Zellner, Amsterdam: North-Holland/Elsevier, pp 233-243.

18. George EI, Foster DP (2000) Calibration and empirical Bayes variable selection. Biometrika 87: 731-747.

19. Fernández C, Ley E, Steel MFJ (2001) Benchmark priors for Bayesian model averaging. J of Econometrics 100: 381-427.

20. Cai A, Tsay RS, Chen R (2007) Variable selection in linear regression with many predictors. Technical Report, University of Illinois at Chicago.

21. Lan H, Chen M, Flowers JB, Yandell BS, Stapleton DS, et al. (2006) Combined expression trait correlations and expression quantitative trait locus mapping. PLoS Genetics 2: e6.

22. Zhang D, Lin Y, Zhang M (2009) Penalized orthogonal-components regression for large *P* small *n* data. Electron J of Statist 3: 781-796.

This article was originally published in a special issue, **Statistical Methods: Markov Chain Monte Carlo** handled by Editor(s). Dr. Faming Liang, Texas A&M University, USA; Dr.Nengjun Yi, University of Alabama at Birmingham, USA