We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

10

Annealing Stochastic Approximation Monte Carlo for Global Optimization

Faming Liang
Department of Statistics Texas A&M University
USA

1. Introduction

During the past several decades, simulated annealing (Kirkpatrick *et al.*, 1983) and the genetic algorithm (Holland, 1975; Goldberg, 1989) have been applied successfully by many authors to highly complex optimization problems in different fields of sciences and engineering. In spite of their successes, both algorithms suffer from some difficulties in convergence to the global optima.

Suppose that we are interested in minimizing the function $U(x)$ over a given space X. Throughout this article, $U(x)$ is called the energy function in terms of physics. Simulated annealing works by simulating a sequence of distributions defined as

$$f_k(x) = \frac{1}{Z_k} \exp\{-U(x)/\tau_k\}, \quad x \in \mathcal{X}, \quad k = 1, 2, \ldots,$$

where τ_k is called the temperature. The temperatures form a decreasing ladder $\tau_1 > \ldots > \tau_k > \ldots$ with τ_1 being reasonably large such that the Metropolis-Hastings (MH) moves (Metropolis *et al.*, 1953; Hastings, 1970) have a high acceptance rate at this level and $\lim_{k\to\infty} \tau_k = 0$. It has been shown by Geman and Geman (1984) that the global minima of U(x) can be reached by simulated annealing with probability 1 if the temperature decreases at a logarithmic rate. In practice, this cooling schedule is too slow; that is, CPU times can be too long to be affordable in challenging problems. Most frequently, people use a linearly or geometrically decreasing cooling schedule, which can no longer guarantee the global minima to be reached.

The genetic algorithm solves the minimization problem by mimicking the natural evolutionary process. A population of candidate solutions (also known as individuals), generated at random, are tested and evaluated for their energy values; the best of them are then bred through mutation and crossover operations; the process repeated over many generations, until an individual of satisfactory performance is found. The mutation operation is modeled by random perturbations of the individuals. The crossover operation is modeled by random perturbations of the couples formed by two individuals selected according to some procedure, e.g., a roulette wheel selection or a random selection. Through the crossover operation, the solution information distributed across the population can be effectively used in the minimization process. Schmitt (2001) showed that under certain conditions, the genetic algorithm can converge asymptotically to the global minima at a logarithmic rate in analogy to simulated annealing.

Source: Simulated Annealing, Book edited by: Cher Ming Tan, ISBN 978-953-7619-07-7, pp. 420, February 2008, I-Tech Education and Publishing, Vienna, Austria

Quite recently, the stochastic approximation Monte Carlo (SAMC) algorithm (Liang *et al.*, 2007) has been proposed in the literature as a dynamic importance sampling technique. A remarkable feature of SAMC is that it possesses the self-adjusting mechanism and is thus not trapped by local energy minima. In this article, we consider applications of SAMC in optimization. Two modified versions of SAMC, annealing SAMC (Liang, 2007; Zhu *et al.*, 2007) and annealing evolutionary SAMC (Liang, 2008), are discussed. Both algorithms have inherited self-adjusting ability from the SAMC algorithm. The annealing SAMC algorithm works in the same spirit as the simulated annealing algorithm but with the sample space instead of temperature shrinking with iterations. The annealing evolutionary SAMC algorithm represents a further improvement of annealing SAMC by incorporating some crossover operators originally used by the genetic algorithm into its search process. Under mild conditions, both annealing SAMC and annealing evolutionary SAMC can converge weakly toward a neighboring set of global minima in the space of energy. The new algorithms are tested on two optimization problems with comparisons with simulated annealing and the genetic algorithm. The numerical results favor to the new algorithms.

The remainder of this article is organized as follows. In Section 2, w describe the ASAMC and AESAMC algorithms and study their convergence. In Section 3, we illustrate the new algorithms with two function minimization problems, one has a rugged energy landscape and the other is a complex least square estimation problem encountered in economic research. In Section 4, we conclude the paper with a brief discussion.

2. Annealing stochastic approximation Monte Carlo algorithms

2.1 Stochastic approximation Monte Carlo

Before describing the annealing SAMC algorithms, we first give a brief description of SAMC. In our description, some of the conditions have been slightly modified from those given in Liang *et al.* (2007) to make the algorithm more suitable for solving optimization problems. Suppose that we are working with the following Boltzmann distribution,

$$f(x) = \frac{1}{Z} \exp\{-U(x)/\tau\}, \quad x \in \mathcal{X}, \tag{1}$$

where Z is the normalizing constant, τ is the temperature, and $\mathcal{X}$ is the sample space. For the reason of mathematical simplicity, we assume that $\mathcal{X}$ is compact. This assumption is reasonable, because for optimization problems we are usually only interested in the optimizers instead of samples drawn from the whole sample space and we have often some rough ideas where the optimizers are. Furthermore, we suppose that the sample space has been partitioned according to the energy function into m disjoint subregions denoted by $E_1 = \{x : U(x) \le u_1\}$, $E_2 = \{x : u_1 < U(x) \le u_2\}$, ..., $E_{m-1} = \{x : u_{m-2} < U(x) \le u_{m-1}\}$, and $E_m = \{x : U(x) > u_{m-1}\}$, where $u_1, \ldots, u_{m-1}$ are real numbers specified by the user. Let $\psi(x)$ be a non-negative function defined on the sample space with $0 < \int_{\mathcal{X}} \psi(x)dx < \infty$, and $g_i = \int_{E_i} \psi(x)dx$. In practice, we often set $\psi(x) = \exp\{-U(x)/\tau\}$.

SAMC seeks to draw samples from each of the subregions with a pre-specified frequency. If this goal can be achieved, then the local-trap problem can be avoided successfully. Let $x^{(t+1)}$

denote a sample drawn from a MH kernel $K_{\theta^{(t)}}(x^{(t)}, \cdot)$ with the proposal distribution $q(x^{(t)}, \cdot)$ and the stationary distribution

$$f_{\theta^{(t)}}(x) \propto \sum_{i=1}^{m} \frac{\psi(x)}{e^{\theta_i^{(t)}}} I(x \in E_i), \tag{2}$$

where $\theta^{(t)} = (\theta_1^{(t)}, \ldots, \theta_m^{(t)})$ is an m-vector in a space Θ.

Let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_m)$ be an m-vector with $0 < \pi_i < 1$ and $\sum_{i=1}^{m} \pi_i = 1$, which defines a desired sampling frequency for the subregions. Henceforth, π will be called the desired sampling distribution. Define $H(\theta^{(t)}, x^{(t+1)}) = (e^{(t+1)} - \pi)$, where $e^{(t+1)} = (e_1^{(t+1)}, \ldots, e_m^{(t+1)})$ and $e_i^{(t+1)} = 1$ if $x^{(t+1)} \in E_i$ and 0 otherwise. Let $\{\gamma_t\}$ be a positive non-decreasing sequence satisfying the conditions,

$$(i) \quad \sum_{t=0}^{\infty} \gamma_t = \infty, \qquad (ii) \quad \sum_{t=0}^{\infty} \gamma_t^{\delta} < \infty, \tag{3}$$

for some $\delta \in (1, 2)$. In this article, we set

$$\gamma_t = \left(\frac{t_0}{\max(t_0, t)} \right)^{\eta} \tag{4}$$

for some specified values of $t_0 > 1$ and $\eta \in (\frac{1}{2}, 1]$. A large value of t_0 will allow the sampler to reach all subregions very quickly even for a large system. Let $J(x)$ denote the index of the subregion the sample x belongs to. With above notations, one iteration of SAMC can be described as follows.

SAMC algorithm:

i. Generate $x^{(t+1)} \sim K_{\theta^{(t)}}(x^{(t)}, \cdot)$ with a single Metropolis-Hastings simulation step:

   (*i.1*) Generate y according to the proposal distribution $q(x^{(t)}, y)$.

   (*i.2*) Calculate the ratio

$$r = e^{(\theta_{J(x^{(t)})}^{(t)} - \theta_{J(y)}^{(t)})} \frac{\psi(y)}{\psi(x^{(t)})} \frac{q(y, x^{(t)})}{q(x^{(t)}, y)}.$$

   (i.3) Accept the proposal with probability $\min(1, r)$. If it is accepted, set $x^{(t+1)} = y$; otherwise, set $x^{(t+1)} = x^{(t)}$.

ii. Set $\theta^* = \theta^{(t)} + \gamma_t H(\theta^{(t)}, x^{(t+1)})$, where γ_t is called the gain factor.

iii. If $\theta^* \in \Theta$, set $\theta^{(t+1)} = \theta^*$; otherwise, set $\theta^{(t+1)} = \theta^* + \boldsymbol{c}^*$, where $\boldsymbol{c}^* = (c^*, \ldots, c^*)$ and c^* is chosen such that $\theta^* + \boldsymbol{c}^* \in \Theta$.

A remarkable feature of ASAMC is its self-adjusting mechanism. If a proposal is rejected, the weight of the subregion that the current sample belongs to will be adjusted to a larger value, and thus the proposal of jumping out from the current subregion will less likely be rejected in the next iteration. This mechanism warrants that the algorithm will not be

trapped by local minima. This is very important for the systems with multiple local energy minima.

The parameter space Θ is set to $[-B_\Theta, B_\Theta]^m$ with B_Θ being a huge number e.g., 10^{100}, which, as a practical matter, is equivalent to setting $B = \mathbb{R}^m$. In theory, this is also fine. As implied by Theorem 5.4 of Andrieu *et al.* (2005), the varying truncation of θ^* can only occur a finite number of times, and thus $\{\theta^{(t)}\}$ can be kept in a compact space during simulations. Note that $f_{\theta^{(t)}}(x)$ is invariant with respect to a location transformation of $\theta^{(t)}$—that is, adding to or subtracting a constant vector from $\theta^{(t)}$ will not change $f_{\theta^{(t)}}(x)$.

The proposal distribution $q(x, y)$ used in the MH moves satisfies the minorisation condition, i.e.,

$$\sup_{\theta\in\Theta}\sup_{x,y\in\mathcal{X}}\frac{f_{\theta^{(t)}}(y)}{q(x,y)}<\infty. \tag{5}$$

The minorisation condition is a natural condition in study of MCMC theory (Mengersen and Tweedie, 1996). In practice, this kind of proposals can be easily designed for both discrete and continuous problems. Since both Θ and $\mathcal{X}$ are compact, a sufficient design for the minorisation condition is to choose $q(x, y) > 0$ for all $x, y \in \mathcal{X}$. For example, for a continuous problem, $q(x, y)$ can be chosen as a random walk Gaussian proposal $y \sim N(x, \sigma^2)$ with σ^2 being calibrated to have a desired acceptance rate. Issues on implementation of the algorithm, such as how to partition the sample space, how to choose the gain factor sequence, and how to set the number of iterations, have been discussed at length in Liang *et al.* (2007).

SAMC falls into the category of stochastic approximation algorithms (Benveniste et al.,1990; Andrieu et al., 2005). The convergence of this algorithm can be extended from a theorem presented in Liang *et al.* (2007). Under mild conditions, we have

$$\theta_i^{(t)}\to\begin{cases}C+\log\left(\int_{E_i}\psi(x)dx\right)-\log\left(\pi_i+\pi_0\right), & \text{if } E_i\neq\emptyset,\\ -\infty. & \text{if } E_i=\emptyset,\end{cases} \tag{6}$$

as $t \to \infty$, where C is an arbitrary constant, $\pi_0 = \sum_{j\in\{i:E_i=0\}}\pi_j/(m-m_0)$, and $m_0 = \#\{i : E_i = ;\}$ is the number of empty subregions. A subregion E_i is called empty if $\int_{E_i}\psi(x)dx = 0$. In SAMC, the sample space partition can be made blindly by simply specifying some values of $u_1, \ldots, u_{m-1}$. This may lead to some empty subregions. The constant C can be determined by imposing a constraint on $\theta^{(t)}$, say, $\sum_{i=1}^{m} e^{\theta_i^{(t)}}$ is equal to a known number.

Let $\hat{\pi}_i^{(t)} = P(x^{(t)} \in E_i)$ be the probability of sampling from the subregion E_i at iteration t. Equation (6) implies that as $t \to \infty$, $\hat{\pi}_i^{(t)}$ will converge to $\pi_i + \pi_0$ if $E_i \neq 0$ and 0 otherwise. With an appropriate specification of $\boldsymbol{\pi}$, sampling can be biased to the low energy subregions to increase the chance of finding the global energy minima.

The subject of stochastic approximation was founded by Robbins and Monro (1951). After five decades of continual development, it has developed into an important area in systems control and optimization. Many of the neural network training algorithms, such as the simultaneous perturbation stochastic approximation algorithm (Spall, 1992), the Widrow Hoff algorithm (also known as the "least mean square" algorithm) (Haykin, 1999, pp.128-135), the Alopex algorithm (Harth & Tzanakou, 1974) and self-organizing maps (Kohonen, 1990), can be regarded as special instances of stochastic approximation. Refer to Bharath & Borkar (1999) for more discussions on this issue. Recently, stochastic approximation has been used with Markov chain Monte Carlo for solving maximum likelihood estimation problems (Gu & Kong, 1998; Delyon *et al.*, 1999). The critical difference between SAMC and other stochastic approximation algorithms is at sample space partitioning. Sample space partitioning improves the performance of stochastic approximation in optimization. It forces each non-empty subregion to be visited with a fixed frequency, and thus increases the chance to locate the global energy minimizer.

2.2 Annealing stochastic approximation Monte Carlo

Like conventional Markov chain Monte Carlo algorithms, SAMC is able to find the global energy minima if the run is long enough. However, due to the broadness of the sample space, the process may be slow even when sampling has been biased to low energy subregions. To accelerate the search process, we shrink the sample space over iterations. As argued below, this modification preserves the theoretical property of SAMC when a global proposal distribution is used.

Suppose that the subregions $E_1, \ldots, E_m$ have been arranged in ascending order by energy; that is, if $i < j$, then $U(x) < U(y)$ for any $x \in E_i$ and $y \in E_j$. Let $\varpi(u)$ denote the index of the subregion that a sample x with energy u belongs to. For example, if $x \in E_j$, then $\varpi(U(x)) = j$. Let $\mathcal{X}^{(t)}$ denote the sample space at iteration t. Annealing SAMC initiates its search in the entire sample space $\mathcal{X}_0 = \bigcup_{i=1}^{m} E_i$, and then iteratively searches in the set

$$\mathcal{X}_t = \bigcup_{i=1}^{\varpi(U_{\min}^{(t)}+\aleph)} E_i, \quad t = 1, 2, \ldots, \tag{7}$$

where $U_{\min}^{(t)}$ is the best energy value obtained until iteration t, and $\aleph > 0$ is a user specified parameter which determines the broadness of the sample space at each iteration. Since the sample space shrinks iteration by iteration, the algorithm is called annealing SAMC. In summary, the ASAMC algorithm consists of the following steps:

ASAMC algorithm:

a) (Initialization) Partition the sample space $\mathcal{X}$ into m disjoint subregions $E_1, \ldots, E_m$ according to the objective function $U(x)$; specify a desired sampling distribution $\boldsymbol{\pi}$; initialize $x^{(0)}$ by a sample randomly drawn from the sample space $\mathcal{X}$, $\theta^{(0)} = (\theta_1^{(0)}, \ldots, \theta_m^{(0)}) = (0, 0, \ldots, 0)$, $\aleph$, and $\mathcal{X}_0 = \bigcup_{i=1}^{m} E_i$; and set the iteration number $t = 0$. Let Θ be a compact subset in R^m, $\theta^{(0)} = (\theta_1^{(0)}, \ldots, \theta_m^{(0)}) = (0, 0, \ldots, 0)$, and $\theta^{(0)} \in \Theta$.

b) (Sampling) Update the current sample $x^{(t)}$ by a single or few MH moves which admit the following distribution as the invariant distribution,

$$f_{\theta^{(t)}}(x) \propto \sum_{i=1}^{\varpi(U_{\min}^{(t)}+\aleph)} \frac{\psi(x)}{e^{\theta_i^{(t)}}} I(x \in E_i), \tag{8}$$

where $I(x \in E_i)$ is the indicator function, $\theta^{(t)}\, i$ is a working weight associated with the subregion E_i, $\psi(x) = \exp\{-U(x)/\tau\}$ is an unnormalized Boltzmann density, and τ is a user-specified parameter. Denote the new sample by $x^{(t+1)}$.

c) (Working weight updating) Update the working weight $\theta^{(t)}$ as follows:

$$\theta_i^* = \theta_i^{(t)} + \gamma_{t+1}\left[I(x^{(t+1)} \in E_i) - \pi_i\right], \quad i = 1, \ldots, \varpi(U_{\min}^{(t)} + \aleph),$$

where γ_t is called the gain factor. If $\theta^* \in \Theta$, set $\theta^{(t+1)} = \theta^*$; otherwise, set $\theta^{(t+1)} = \theta^* + \boldsymbol{c}^*$, where $\boldsymbol{c}^* = (c^*, \ldots, c^*)$ and c^* is chosen such that $\theta^* + \boldsymbol{c}^* \in \Theta$

d) (Termination Checking) Check the termination condition, e.g., a fixed number of iterations or an optimal solution set have been reached. Otherwise, set $t \leftarrow t + 1$ and go to step (b).

It has been shown in Liang (2007) that if the gain factor sequence satisfie (3) and the proposal distribution satisfies the minorisation condition (5), ASAMC can converge weakly toward a neighboring set of the global minima of $U(x)$ in the space of energy. More precisely, the sample $x^{(t)}$ converges in distribution to a random variable with the density function

$$f_\theta(x) \propto \sum_{i=1}^{\varpi(u_{\min}+\aleph)} \frac{\pi_i' \psi(x)}{\int_{E_i} \psi(x)dx} I(x \in E_i), \tag{9}$$

where u_{min} is the global minimum value of $U(x)$, $\pi_i' = \pi_i + (1 - \sum_{j \in \{k: E_k \neq 0, k=1,\ldots,\varpi(u_{min}+\aleph)\}} \pi_i)/m_0$, m_0 is the cardinality of the set $\{k : Ek \neq 0, k = 1, \ldots, \varpi(u_{min} + \aleph)\}$, and 0 denotes an empty set. The subregion E_i is called empty if $\int_{E_i} \psi(x)dx = 0$. An inappropriate specification of u_i's may result in some empty subregions. ASAMC allows for the existence of empty subregions in simulations.

2.3 Annealing evolutionary stochastic approximation Monte Carlo

Like the genetic algorithm, AESAMC also works on a population of samples. Let $x = (x_1, \ldots, x_n)$ denote the population, where n is the population size, and $x_i = (x_{i1}, \ldots, x_{id})$ is a d-dimensional vector called an individual or chromosome in terms of genetic algorithms. Thus, the minimum of $U(x)$ can be obtained by minimizing the function $U(x) = \sum_{i=1}^{n} U(x_i)$. An unnormalized Boltzmann density can be defined for the population as follows,

$$\boldsymbol{\psi}(\boldsymbol{x}) = \exp\{-\boldsymbol{U}(\boldsymbol{x})/\tau\}, \quad \boldsymbol{x} \in \mathcal{X}^n, \tag{10}$$

where $\mathcal{X}_n = \mathcal{X} \times \cdots \times \mathcal{X}$ is a product sample space. The sample space can be partitioned according to the function $U(x)$ into m subregions: $\mathbb{E}_1 = \{x : U(x) \le u_1\}$, $\mathbb{E}_2 = \{x : u_1 < U(x) \le u_2\}$, . . ., $\mathbb{E}_{m-1} = \{x : u_{m-2} < U(x) \le u_{m-1}\}$, and $\mathbb{E}_m = \{x : U(x) > u_{m-1}\}$, where $u_1, \ldots, u_{m-1}$ are $m - 1$ known real numbers. As in ASAMC, we suppose that the subregions have been arranged in ascending order by the function $U(x)$. We note that here the sample space is not necessarily partitioned according to the function $U(x)$, for example, $\lambda(x) = \min\{U(x_1), \ldots, U(x_n)\}$ is also a good choice. The population can then evolve under the framework of ASAMC with an appropriate specification of the proposal distribution for the MH moves. At iteration t, the MH moves admit the following distribution as the invariant distribution,

$$\boldsymbol{f}_{\theta^{(t)}}(\boldsymbol{x}) \propto \sum_{i=1}^{\varpi(\boldsymbol{U}_{\min}^{(t)}+\aleph)} \frac{\boldsymbol{\psi}(\boldsymbol{x})}{e^{\theta_i^{(t)}}} I(\boldsymbol{x} \in \mathbb{E}_i), \quad \boldsymbol{x} \in \mathcal{X}_t^n, \tag{11}$$

where $U_{\min}^{(t)}$ denotes the best value of $U(x)$ obtained by iteration t. As discussed before, $\{\theta^{(t)}\}$ can be kept in a compact space in simulations.

Since, in AESAMC, the state of the MH chain has been augmented to a population, the crossover operators used in the genetic algorithm can be employed to accelerate the evolution of the population. However, to satisfy the Markov chain reversibility condition, these operators need to be modified appropriately. As demonstrated by Liang and Wong (2000, 2001), Goswami and Liu (2007), and Jasra *et al.* (2007), incorporating genetic type moves into Markov chain Monte Carlo can often improve the mixing rate of the simulation. Note that the crossover operators used in these work may not be suitable for AESAMC, because asymptotic independence between individuals is required to be remained in the operations. Whilst this is not required in AESAMC. The sample space partition makes the individuals dependent on each other, although $\psi(x)$ is defined as a product of the functions $\psi(x_i)$, $i = 1, \ldots, n$. The crossover operators used in AESAMC can be described as follows.

K-Point Crossover This proposal is the same as that used in Liang and Wong (2001). To make the article self-contained, it is briefly described as follows. Two chromosomes, say x_i and x_j with $i < j$, are selected from the current population x according to some selection procedure as parental chromosomes and two offspring chromosomes x'_i and x'_j are generated as follows: sample K integer crossover points without replacement from the set $\{1, \ldots, d-1\}$; sort these points in ascending order; and construct offspring chromosomes by swapping the genes of the parental chromosomes between each odd and the next even crossover points (d is set as an additional crossover point when K is odd). The new population can then be formed as $x' = (x_1, \ldots, x_{i-1}, x'_i, x_{i+1}, \ldots, x_{j-1}, x'_j, x_{j+1}, \ldots, x_n)$. In this article, 1 and 2-point crossover operators are applied equally when the K point crossover operator is selected in simulations. An extreme case of the K-point crossover is the uniform crossover, in which each element of x'_i (i.e., genotype) is randomly chosen from the two corresponding elements of the parental chromosomes and the corresponding element of x_j is assigned to the element not chosen by x_{0i}.

Throughout this article, the parental chromosomes for the *K*-point crossover operator are selected as follows. The first parental chromosome is selected from the current population according to the distribution

$$w_1(x) = \frac{\exp\{-U(x)/\tau_s\}}{\sum_{k=1}^{n} \exp\{-U(x_k)/\tau_s\}}, \quad x \in \{x_1, \ldots, x_n\}, \tag{12}$$

where τ_s is the selection temperature. In this article, we set $\tau_s = \tau / 10$. Let x_i denote the first parental chromosome. The second parental chromosome is then selected from the subpopulation $\boldsymbol{x} \setminus \{x_i\}$ according to the distribution

$$w_2(x|x_i) = \frac{\exp\{-U(x)/\tau_s\}}{\sum_{k\neq i} \exp\{-U(x_k)/\tau_s\}}, \quad x \in \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}. \tag{13}$$

For a given pair of chromosomes (x_i, x_j), the selection probability is $P_s(x_i, x_j \,|\boldsymbol{x}) = w_1(x_i)w_2(x_j \,|\, x_i) + w_1(x_j)w_2(x_i \,|\, x_j)$.

To have the distribution (11) invariant with respect to the crossove operation, the acceptance of the new population x' should be moderated by the MH rule; that is, accepting $x'$with probability

$$\min\left\{1, [\boldsymbol{f}_{\theta^{(t)}}(\boldsymbol{x}')T(\boldsymbol{x}' \to \boldsymbol{x})]/[\boldsymbol{f}_{\theta^{(t)}}(\boldsymbol{x})T(\boldsymbol{x} \to \boldsymbol{x}')]\right\}, \tag{14}$$

where the transition probability ratio is

$$T(\boldsymbol{x}' \to \boldsymbol{x})/T(\boldsymbol{x} \to \boldsymbol{x}') = P_s(x_i', x_j'|\boldsymbol{x}')/P_s(x_i, x_j|\boldsymbol{x}).$$

Snooker Crossover This operator proceeds as follows:

a) Randomly select one chromosome, say x_i, from the current population $\boldsymbol{x}$.
b) Select the other chromosome, say, x_j , from the subpopulation $\boldsymbol{x} \setminus \{x_i\}$ according to the distribution $w_2(x \,|\, x_i)$ as defined in (13).
c) Let $e = (x_j - x_i)/\|x_j - x_i\|$, and let $x'_i = x_i + re$, where r is a random variable drawn for a normal distribution $N(0, \sigma_c^2)$ with the standard deviation σ_c being calibrated such that the operation has a reasonable overall acceptance probability.
d) Construct a new population $\boldsymbol{x}'$ by replacing x_i with the offspring x'_i , and accept the new population with probability $\min\{1, \boldsymbol{f}_{\theta^{(t)}}(\boldsymbol{x}')/\boldsymbol{f}_{\theta^{(t)}}(\boldsymbol{x})\}$. For this operator, the transition probability ratio is 1; that is, $T(\boldsymbol{x} \to \boldsymbol{x}') = T(x' \to x)$.

We note that this operator is a little different from the the snooker sampler described in Gilks *et al.* (1994) and the snooker crossover operator described in Liang and Wong (2001), where x_i is required to be independent of other individuals and the operation is required to leave the marginal distribution of x_i invariant. These requirements are waived here because AESAMC allows for the dependence among the individuals.

Linear Crossover This operator has the same procedure with the snooker crossover operator except that step (c) is changed as follows.

(c) Set $x'_i = x_i + rx_j$, where r ~ Unif [−1, 1].

This operator is designed for exploration of the triangular region between x_i and x_j and that between x_i and $-x_j$.

Mutation In addition to the crossover operators, AESAMC also needs some mutation operators. Since the population size can be large, we here assume that the population is updated in the style of the Gibbs sampler, individual by individual, or component by component by viewing the population as a long vector. Furthermore, we assume that the mutation operator satisfies the minorisation condition

$$\sup_{\theta\in\Theta}\sup_{x\in\mathcal{X}}\frac{\boldsymbol{f}_\theta(x_i',\boldsymbol{x}_{[-i]})}{q(x_i\to x_i'|\boldsymbol{x}_{[-i]})}<\infty,\quad\forall\ i=1,\ldots,n, \tag{15}$$

where $\boldsymbol{x}_{[-i]} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ and $q(x_i \to x'_i \,|\boldsymbol{x}_{[-i]})$ denotes the proposal distribution use for updating x_i. Note that the condition (15) can be further relaxed for AESAMC, requiring the inequality holds for any components of x_i. Since both $\mathcal{X}$ and Θ are compact, it is easy to verify that the mutation operators described below satisfy the minorisation condition.

Two types of mutation operators, MH-Gibbs mutation and point mutations, are used in this article. The MH-Gibbs mutation, also called the "Metropolis-within-Gibbs" sampler (Müller, 1991) or the hybrid MCMC sampler (Robert & Casella, 2004, pp.393), proceeds as follows.

For $i = 1, \ldots, n$, given the population $\boldsymbol{x}^{(t+1,i-1)} = (x_1^{(t+1)}, \ldots, x_{i-1}^{(t+1)}, x_i^{(t)}, \ldots, x_n^{(t)})$:

a) Generate a random direction vector e from a uniform distribution on the surface of a unit d-dimensional hypersphere.

b) Generate a random number $r \sim N(0, \sigma_m^2)$ and set $x'_i = x_i^{(t)} + re$, where σ_m is calibrated such that the operator has a reasonable overall acceptance rate.

c) Construct a new population by replacing $x_i^{(t)}$ with x'_i, accept the new population according to the MH rule as in the snooker crossover, and denote the new population by $\boldsymbol{x}^{(t+1,i)}$.

For the reason of mathematical simplicity, we keep the working weights $(\theta_1^{(t)}, \ldots, \theta_m^{(t)})$ unchanged in the cycle of the MH-Gibbs mutation. If the vector e used in step (b) is a (0,1)-binary vector with the positions of nonzero elements being selected randomly, the above operator is called the point mutation operator. If the total number of nonzero elements in e is K, then the operator is called the K-point mutation operator. In this article, 1 and 2-point mutation operators are applied equally when the K-point mutation operator is selected in simulations. Let σ_p denote the step size of the K-point mutation; that is, the operator can be expressed as $x'_{ij} = x_{ij}^{(t)} + r_j e_j$ for $j = 1, \ldots, d$, where r_j is a random number drawn from the normal $N(0, \sigma_p^2)$ and σ_p is called the step size of the operator.

AESAMC Algorithm Let $\rho_1, \ldots, \rho_5$, $0 < \rho_i < 1$ and $\sum_{i=1}^{5} \rho_i = 1$, denote the respective probabilities for the MH-Gibbs mutation, K-point mutation, K point crossover, snooker crossover, and linear crossover operators to work at each iteration. In this article, we set $\rho_1 = \rho_2 = 0.05$ and $\rho_3 = \rho_4 = \rho_5 = 0.3$. The AESAMC algorithm can be summarized as follows.

AESAMC algorithm:

a) (Initialization) Partition the sample space $\mathcal{X}_n$ into m disjoint subregions $\mathbb{E}_1, \ldots, \mathbb{E}_m$; choose the threshold value $\aleph$ and the working probabilities $\rho_1, \ldots, \rho_5$; initialize a population $\boldsymbol{x}^{(0)}$ at random; and set $\theta^{(0)} = (\theta_1^{(0)}, \ldots, \theta_m^{(0)}) = (0, 0, \ldots, 0)$, $\mathcal{X}_0^n = \bigcup_{i=1}^m \mathbb{E}_i$, $\boldsymbol{U}_{\min}^{(0)} = \boldsymbol{U}(\boldsymbol{x}^{(0)})$ and $t = 0$.

b) (Sampling) Update the current population $\boldsymbol{x}^{(t)}$ using the MH-Gibbs mutation, K-point mutation, K-point crossover, snooker crossover, and linear crossover operators according to the respective working probabilities.

c) (Working weight updating) Update the working weight $\theta^{(t)}$ by setting

$$\theta_i^* = \theta_i^{(t)} + \gamma_{t+1} H_i(\theta^{(t)}, \boldsymbol{x}^{(t+1)}), \quad i = 1, \ldots, \varpi(\boldsymbol{U}_{\min}^{(t)} + \aleph),$$

where $H_i(\theta^{(t)}, \boldsymbol{x}^{(t+1)}) = I(\boldsymbol{x}^{(t+1)} \in \mathbb{E}_i) - \pi_i$ for the crossover operators, and $H_i(\theta^{(t)}, \boldsymbol{x}^{(t+1)}) = \sum_{j=1}^n I(\boldsymbol{x}^{(t+1,j)} \in \mathbb{E}_i)/n - \pi_i$ for the mutation operators. If $\theta^* \in \Theta$, set $\theta^{(t+1)} = \theta^*$; otherwise, set $\theta^{(t+1)} = \theta^* + \boldsymbol{c}^*$, where $\boldsymbol{c}^* = (c^*, \ldots, c^*)$ and c^* is chosen such that $\theta^* + \boldsymbol{c}^* \in \Theta$.

d) (Termination Checking) Check the termination condition, e.g., a fixed number of iterations or an optimal solution set have been reached. Otherwise, set $t \to t + 1$ and go to step (b).

It has been shown in Liang (2008) that if the mutation operator satisfies the minorisation conditio (5) and the gain factor sequence satisfies (3), AESAMC also converges weakly toward a neighboring set of global minima of $\boldsymbol{U}(\boldsymbol{x})$ in the space of energy.

2.4 Practical issues

Liang *et al.* (2007) discussed practical issues on implementation of SAMC Some rules developed there, e.g., those on sample space partitioning and convergence diagnostic, are still applicable to ASAMC and AESAMC. Briefly speaking, the sample space should be partitioned such that the MH moves within the same subregion have a reasonable acceptance rate. In this article, the sample space is partitioned such that each subregion has an equal energy bandwidth Δu, i.e., $u_{i+1} - u_i \equiv \Delta u$ for all $i = 1, \ldots, m - 1$. To ensure that the moves within the same subregion have a reasonable acceptance rate, it is set $\Delta u = 0.2\tau$. The convergence can be diagnosed by examining the difference of the patterns of the working weights obtained in multiple runs. In the below, we discuss three more issues related to ASAMC and AESAMC.

- On the choice of $\boldsymbol{\pi}$. Liang *et al.* (2007) suggest to set $\boldsymbol{\pi}$ biased to the low energy regions if one aims at minimization. However, this is different for ASAMC and AESAMC, as it includes an extra parameter, namely, $\aleph$, to control its search space at each iteration. In our experience, a uniform setting is often better than a low-energy biasing setting in terms of efficiency of the two algorithms, especially when $\aleph$ is small. Under the uniform setting, the system has more chances to visit higher energy regions, and thus has more chances to transit between disconnected regions. In this chapter, we set $\boldsymbol{\pi}$ to be uniform over all subregions, i.e., $\pi_1 = \ldots = \pi_m = 1/m$, in all computations.
- On the choice of $\aleph$, T_0 and N, where N denotes the total number of iterations. Since $\aleph$ determines the size of the neighboring set toward which ASAMC and AESAMC

converge, ℵ should be chosen carefully for better efficiency of the algorithms. If ℵ is too small, it may take a long time for the algorithms to locate the global minima. In this case, the sample space may contain a lot of separated regions, and most of the proposed transitions will be rejected if the proposal distribution is not spread enough. If ℵ is too large, it may also take a long time for the algorithms to locate the global energy minimum due to the broadness of the sample space. In principle, the value of ℵ should be chosen according to the roughness of the energy function. The rougher the energy function is, the larger value of ℵ one should choose. A large value of ℵ should associate with a large value of T_0, as a larger value of T_0 means faster transitions over the entire sample space. In practice, the values of ℵ, T_0 and N can be determined through a trial and error process based on the diagnosis for the convergence of the algorithms. If they fail to converge, the parameters should be tuned to larger values. Finally, we point out that due to the population effect, AESAMC can often work with a smaller value of ℵ than ASAMC.

- On the choice of population size for AESAMC. The genetic algorithm often works with a large population, because the crossover operation is the key to its efficiency. In AESAMC, the crossover operator has been modified to serve as a proposal for the MH moves, and it is no longer as critical as to the genetic algorithm. In AESAMC, the population size is usually set to a moderate number, ranging from 5 to 50. As known by many people, the crossover operation favors to high dimensional problems.

3. Numerical examples

3.1 A multiple local minima problem

To illustrate ASAMC, we consider minimizing the following function on $[-1.1, 1.1]^2$:

$$U(\boldsymbol{x}) = -\big(x_1 \sin(20x_2)+x_2 \sin(20x_1)\big)^2 \cosh\big(\sin(10x_1)x_1\big)-\big(x_1 \cos(10x_2)-x_2 \sin(10x_1)\big)^2 \cosh\big(\cos(20x_2)x_2\big),$$

whose global minimum is -8.12465 attained at (x_1, x_2) = (−1.0445,−1.0084) and (1.0445,−1.0084) This example is identical to Example 6.1 of Liang (2005). Figure 1 shows that $U(\boldsymbol{x})$ has a multitude of local energy minima separated by high-energy barriers.

Since the dimension of the problem is low, AESAMC was not applied to this example. In applyin ASAMC to this example, we partitioned the sample space into 41 subregions with an equal energy bandwidth: $E_1 = \{\boldsymbol{x} \in \mathcal{X} : U(\boldsymbol{x}) \le -8.0\}$, $E_2 = \{(\boldsymbol{x}) \in \mathcal{X} : -8.0 < U(\boldsymbol{x}) \le -7.8\}$, . . ., and $E_{41} = \{(\boldsymbol{x}) \in \mathcal{X} : -0.2 < U(\boldsymbol{x}) \le 0\}$, set $\psi(\boldsymbol{x}) = \exp(-U(\boldsymbol{x})/0.1)$, $t_0 = 100$, and ℵ = 6, and choose the proposal distribution as $N_2(\boldsymbol{x}, 0.3_2 I_2)$, where I_d denotes an identity matrix of size d by d. The algorithm was run for 50000 iterations, and 500 samples were collected at equally spaced time intervals.

For comparison, SAMC and SA were also applied to this example. SAMC was run for 50000 iterations with the same setting as ASAMC (the parameter ℵ does not exist in SAMC), and 500 samples were collected at equally spaced time intervals. For SA, we tried the linear and geometric cooling schedules.

a) Linear. The temperature decreases linearly, i.e.,

$$T_k = T_{k-1} - \varrho_l,\ k = 1, \ldots, K,$$

where $\varrho_l = (T_1 - T_K)/(K - 1)$.

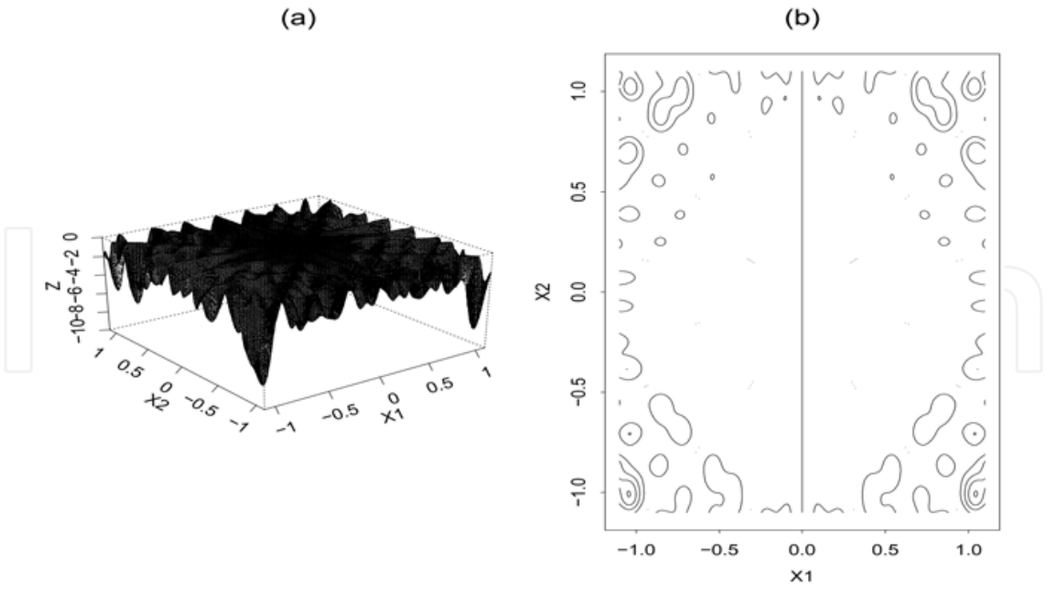


Figure 1. Grid (a) and contour (b) representations of the function $U(\boldsymbol{x})$ on $[-1.1, 1.1]^2$. This figure characterizes multiple local minima problems.

b) Geometric. The temperature decreases geometrically with a constant rate, i.e.,

$$T_k = \varrho_e T_{k-1},\ k = 1, \ldots, K,$$

where $\varrho_e = \exp\{(\log T_K - \log T_1)/(T - 1)\}$.

In all simulations, we set the total number of temperature levels $K = 500$, and set the number of iterations performed at each temperature level to $N_k = N/K$, where N is the total numbers of iterations of a run. For this example, we set $T_1 = 10$, $T_{500} = 0.01$ and $N = 50000$ for both cooling schedules. The resulting values of ϱ_l and ϱ_e are 0.02 and 0.986, respectively. The proposal distribution used at level k is $N(\boldsymbol{x}_t, 0.1^2 T_k I_2)$. In each run, 500 samples were collected at equally spaced time intervals.

Figure 2 shows the evolving paths of the samples collected in the above runs. It is remarkable that ASAMC is ergodic as shown by Figure 2(a). Even though the sample space has been restricted to four isolated regions (four corners) by the choice of $\aleph$, successful transitions between different regions can still be made due to the use of the global proposal distribution. This also explains why a widely spread proposal distribution is preferred in ASAMC. Comparing to the sample path of SAMC, we can see that in ASAMC, sampling is more focused on the low energy regions. Hence, ASAMC is potentially more efficient than SAMC in optimization.

Figures 2(c) and 2(d) show that at high temperatures, SA results in a random walk in the sample space; and that at low temperatures, SA tends to get trapped in a local minimum. Note that the linear cooling schedule contains more high temperature levels than the geometric cooling schedule. The sample paths of SA are significantly different from those of SAMC and ASAMC. The central part of the sample space (Figure 1(b)) has a big area, which is about half of the total area of the sample space, but it is seldom visited by ASAMC and SAMC. However, this part is visited by SA frequently with both linear and geometric cooling schedules. The reason is that SA tends to have a random walk in the sample space at

high temperatures, whereas ASAMC (so is SAMC) tends to have a random walk in the space of subregions, if each subregion is regarded as a single point. This implies that potentially ASAMC can overcome any barriers on the energy landscape and locate global energy minima quickly. Figure 2 shows that during the above runs SAMC and ASAMC have located the global energy minima many times, whilst SA has only located them a few times.

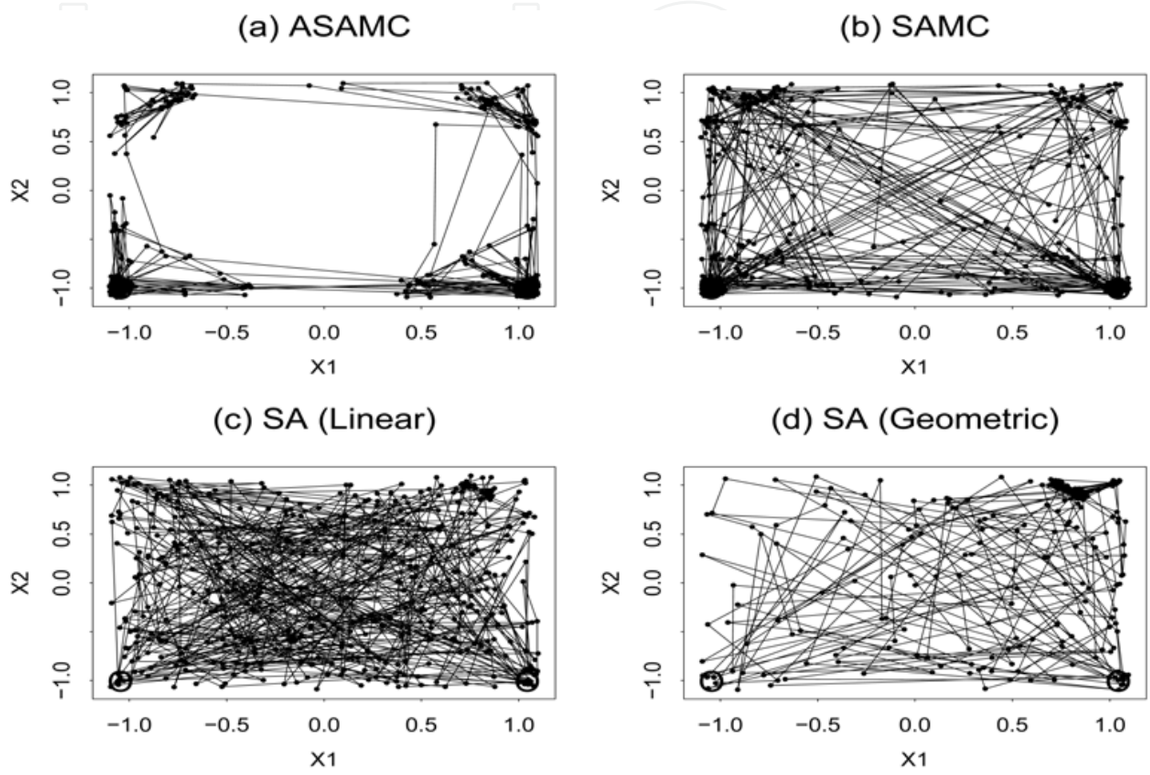


Figure 2. Sample paths of the ASAMC, SAMC and SA samples. The circles in the plots show the locations of the two global energy minima. (a) Sample path of ASAMC. (b) Sample path of SAMC. (c) Sample path of SA with the linear cooling schedule. (d) Sample path of SA with the geometric cooling schedule. This figure characterizes the performance of ASAMC for multiple local minima problems: it is capable of transiting between different local minimum regions.

To compare efficiency of ASAMC, SAMC and SA in global optimization, we conducted the following experiment. Each algorithm was run 1000 times independently. Each run consisted of 20000 iterations. ASAMC and SAMC were run under the same setting as used above except that the proposal distribution was changed to $N_2(\boldsymbol{x}, 0.1^2 I_2)$. This change would force them to move more locally and thus to have more chances to locate the global energy minima. The proposal distribution used in SA has already been fine enough, and was not changed in this experiment. The numerical results are summarized in Table 1. The comparison shows that both ASAMC and SAMC are superior to SA for this example. Note that in all runs of the three algorithms, the total numbers of iterations were the same, and they cost about the same CPU times because the CPU time cost by each iteration is dominated by the part used for energy evaluation. This is especially true for more complicated problems, e.g., the neural network training problems studied in Liang (2007).

Algorithm	Mean	SD($\times 10^{-3}$)	Minimum	Maximum	Proportion
ASAMC	-8.12320	0.047	-8.12465	-8.11278	968
SAMC	-8.12308	0.059	-8.12465	-8.10191	944
SA-1	-8.10326	1.264	-8.12465	-7.77922	572
SA-2	-8.08168	3.102	-8.12466	-7.33146	609

Table 1: Comparison of the SAMC, ASAMC, and SA algorithms for the multiple local minima example. Notations: let z_i denote the minimum energy value obtained in the ith run for $i = 1, \ldots, 1000$, "Mean" is the average of z_i, "SD" is the standard deviation of "Mean", "Minimum"= $\min_{i=1}^{1000} z_i$, "Maximum"= $\max_{i=1}^{1000} z_i$, and "Proportion"= $\#\{i : z_i \leq -8.12\}$. The cooling schedules used in SA-1 and SA-2 are linear and geometric, respectively.

3.2 Rational-expectations model

To illustrate the performance of AESAMC, we consider the following example, which is specified by the system of equations,

$$\begin{aligned} y_{1t} &= g_{1t}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) + \epsilon_{1t}, \\ y_{2t} &= g_{2t}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) + \epsilon_{2t}, \\ x_{t1} &= \gamma_1 x_{t-1,1} + u_{t1}, \\ x_{t2} &= \gamma_2 x_{t-1,2} + u_{t2}, \\ x_{t3} &= \gamma_3 x_{t-1,3} + u_{t3}, \end{aligned} \tag{16}$$

where $\alpha = (\alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{21}, \alpha_{22}, \alpha_{23})$, $\beta = (\beta_{11}, \beta_{12}, \beta_{21}, \beta_{22})$, $\gamma = (\gamma_1, \gamma_2, \gamma_3)$, $\Lambda = [1 - \beta_{12}\beta_{21}(1 - \beta_{11})(1 - \beta_{22})]^{-1}$, and

$$\begin{aligned} g_{1t}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) &= \alpha_{11} x_{t1} + \alpha_{12} x_{t2} + \alpha_{13} x_{t3} + \beta_{11} E_{t-1} y_{t1} + \beta_{12} E_{t-1} y_{t2}, \\ g_{2t}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) &= \alpha_{21} x_{t1} + \alpha_{22} x_{t2} + \alpha_{23} x_{t3} + \beta_{21} E_{t-1} y_{t1} + \beta_{22} E_{t-1} y_{t2}, \\ E_{t-1} y_{t1} &= \frac{\Lambda}{1 - \beta_{11}} \big\{ \alpha_{11}\gamma_1 x_{t-1,1} + \alpha_{12}\gamma_2 x_{t-1,2} + \alpha_{13}\gamma_3 x_{t-1,3} \\ &\quad + \beta_{12}(1 - \beta_{22})^{-1} \big[\alpha_{21}\gamma_1 x_{t-1,1} + \alpha_{22}\gamma_2 x_{t-1,2} + \alpha_{23}\gamma_3 x_{t-1,3} \big] \big\}, \\ E_{t-1} y_{t2} &= \frac{\Lambda}{1 - \beta_{11}} \big\{ \alpha_{21}\gamma_1 x_{t-1,1} + \alpha_{22}\gamma_2 x_{t-1,2} + \alpha_{23}\gamma_3 x_{t-1,3} \\ &\quad + \beta_{21}(1 - \beta_{11})^{-1} \big[\alpha_{11}\gamma_1 x_{t-1,1} + \alpha_{12}\gamma_2 x_{t-1,2} + \alpha_{23}\gamma_3 x_{t-1,3} \big] \big\}. \end{aligned}$$

The parameters (α,β,γ) can be estimated by minimizing the function

$$U(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma},x_{01},x_{02},x_{03}) = \sum_{t=1}^{T}(y_{1t}-g_{1t}(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}))^2 + \sum_{t=1}^{T}(y_{2t}-g_{2t}(\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}))^2 + \sum_{i=1}^{3}\sum_{t=1}^{T}(x_{ti}-\gamma_i x_{t-1,i})^2, \tag{17}$$

on the space: $|\alpha_{ij}| \le 10$, $|\beta_{ij}| \le 10$, $|\gamma_i| \le 1$ and $|x_{0i}| \le 10$.

This example is constructed by Dorsey & Mayer (1995) based on the rational-expectations model encountered in economic research (Hoffman & Schmidt, 1981). Following Dorsey & Mayer (1995), we use a dataset consisting of 40 observations simulated under the specifications: $\alpha_{ij} = 1$ and $\beta_{ij} = 0.2$ for all i and j, $\gamma_1 = 0.1$, $\gamma_2 = 0.2$, $\gamma_3 = 0.8$, $\epsilon_{t1} \sim N(0, 25_2)$, $\epsilon_{t2} \sim N(0, 1)$, $u_{t1} \sim N(0, 36_2)$, $U_{u2} \sim N(0, 4_2)$, and $u_{t3} \sim N(0, 9_2)$. Dorsey and Mayer (1995) assessed the computational difficulty of this problem by running the Marquardt-Levenberg gradient algorithm from 50 different randomly chosen points in the search space. In 49 out of 50 runs, the Marquardt-Levenberg algorithm failed to converge because of either singularities or floating-point overflow, and on the one run that did converge was discovered to be suboptimal.

AESAMC was first applied to this example with the following specifications: the temperature $\tau = 100$, the population size $n = 25$, the mutation step sizes $\sigma_m = \sigma_p = 1.5$, the snooker crossover step size $\sigma_c = 1$, the threshold value $\aleph = 5000$, the gain factor scale $T_0 = 20000$, and total number of iterations $N = 5 \times 10^6$. To examine the performance of AESAMC in a long run, we set N to a large number. The algorithm was run 20 times. On average, each run requires 1.85×10^7 function evaluations and about 180s CUP time on a 2.8GHZ computer. The results are summarized in Table 2 and Figure 3. To assess the robustness of AESAMC to the choices of $\aleph$ and T_0, the algorithm was also run 20 times with $\aleph = 15000$ and $T_0 = 50000$ (keeping the values of other parameters unchanged). As discussed earlier, a large value of $\aleph$ should associate with a large value of T_0. The results suggest that the performance of AESAMC is quite robust to the choice of $\aleph$ and T_0.

For comparison, ASAMC, SA and the genetic algorithm were also applied to this example. ASAMC and SA employed the mutation operators used by AESAMC as their local samplers. For ASAMC, we tried two different settings of ($\aleph$, T_0): (10000, 20000) and (20000, 50000). Under each setting, ASAMC was run 20 times, and each run consists of 1.85×10^7 iterations. For SA, we tried three different choices of the highest temperature: 500, 1000, and 2000. For each choice, SA was also run 20 times. Each run consists of 100 stages, each stage consists of 1.85×10^5 iterations, and the temperature ladder decreases at a rate of 0.95.

The genetic algorithm has many variants, each covering different applications and aspects. The variant we adopted here is the so-called real-coded genetic algorithm (RGA), which is described in Ali *et al.* (2005). RGA includes three free parameters, namely, the population size, the number of individuals to be updated at each generation, and the number of generations. RGA has been tested by Ali *et al.* (2005) on a variety of continuous global optimization problems. The results indicate that it is effective and comparable or favorable to other stochastic optimization algorithms, such as controlled random search (Price, 1983;

Ali and Storey, 1994) and differential evolution (Storn & Price, 1997). For this example, we tried three different settings of population size: 50, 100 and 200. The number of individuals to be updated at each generation was set to one-fifth of the population size, and the number of generations was chosen such that the total number of function evaluations in each run is about 1.85×10^7. Under each setting, RGA was also run 20 times. The results are summarized in Table 2 and Figure 3.

Table 2 shows that the averages of the best function values produced by AESAMC are lower than those produced by other algorithms, although the runs are so long. Figure 3 plots the average progression curves of the best function values produced by these algorithms. RGA performs less well than SA and ASAMC for this example. This implies that this example does not favor to the crossover operations. Even so, AESAMC still significantly outperforms other algorithms. The average of the best function values produced by ASAMC with 1.85×10^7 function evaluations is about the same as that produced by AESAMC with about 6×10^6 function evaluations. This translates to a 3- fold improvement. The results produced by SA and RGA are not comparable to that produced by AESAMC at all. In our experience, high dimensional optimizations usually favor to the crossover operations.

Algorithm	Setting	Minimum	Maximum	Average	SD	Cost
AESAMC	$\aleph = 5000,\ T_0 = 20000$	62694	62984	62815	19	1.85×10^7
	$\aleph = 15000,\ T_0 = 50000$	62694	63078	62813	29	1.85×10^7
ASAMC	$\aleph = 10000,\ T_0 = 20000$	62694	63893	62864	71	1.85×10^7
	$\aleph = 20000,\ T_0 = 50000$	62694	63370	62940	64	1.85×10^7
SA	$\tau_{high} = 500$	62699	63956	63412	84	1.85×10^7
	$\tau_{high} = 1000$	62695	64242	63543	84	1.85×10^7
	$\tau_{high} = 2000$	62700	64779	63501	118	1.85×10^7
RGA	$n = 50$	64585	66703	65140	122	1.9×10^7
	$n = 100$	64579	65299	64822	65	1.9×10^7
	$n = 200$	64579	65288	64861	63	1.9×10^7

Table 2. Comparison of AESAMC, ASAMC, SA and RGA for minimization of the function (17). Let u_i denote the best function value produced in the i-th run. The numbers in the columns of Minimum, Maximum, Average, and SD are calculated, respectively, as follows: $\min_{1 \le i \le 30} u_i$, $\max_{1 \le l \le 30} u_i$, $\sum_{i=1}^{30} u_i/30$, and the standard deviation of the average. Cost: the number of function evaluations in each run.

4. Discussion

In this article, we have described the ASAMC and AESAMC algorithms. A remarkable feature of the two algorithms is that they are not trapped by local energy minima. Under

mild conditions they can converge weakly toward a neighboring set of the global minima in the space of energy. The algorithms were tested on two examples. The numerical results indicate that ASAMC and AESAMC can significantly outperform their competitors, including SAMC, simulating annealing and genetic algorithms, in terms of quality of the solutions obtained with the same CPU time.

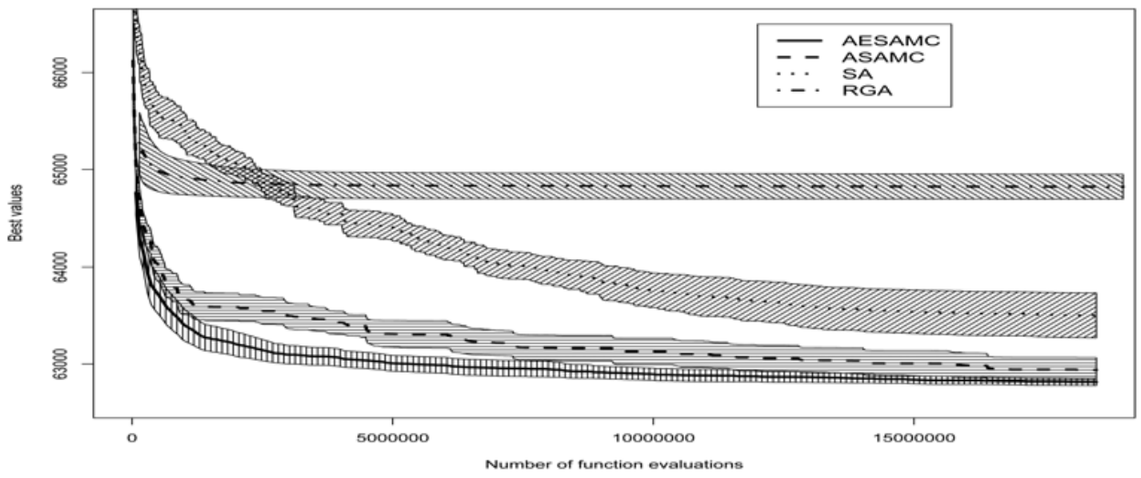


Figure 3: Average progression curves of the best function values (over 20 runs) and their 95% confidence regions (shaded area) produced by AESAMC ($\aleph$= 5000, T_0 = 20000), ASAMC ($\aleph$ = 10000, T_0 = 20000), SA (τ_{high} = 10) and RGA (n = 100) for minimizing the function (17).

In this paper, both ASAMC and AESAMC are described for continuous optimization problems only. Extension to discrete optimization problems is straightforward. For discrete optimization problems, the mutation and crossover operators required by AESAMC can be designed as in Liang & Wong (2000) and Goswami & Liu (2006). The K-point crossover operator described in this paper can also be used for discrete problems.

Although ASAMC and AESAMC are proposed as optimization techniques, they can also be used as importance sampling techniques by keeping the sample space unshrinked through iterations. AESAMC has provided a general framework on how to incorporate crossover operations into dynamically weighted MCMC simulations, e.g., dynamic weighting (Wong & Liang, 1997; Liang, 2002) and population Monte Carlo (Cappé *et al.*, 2004). This framework is potentially more useful than the conventional MCMC framework provided by evolutionary Monte Carlo (Liang & Wong, 2000, 2001). Under the conventional MCMC framework, the crossover operation has often a low acceptance rate. The MH rule will typically reject an unbalanced pair of offspring samples for which one has a high density value and other low. In AESAMC, this difficulty has been much alleviated due to the self-adjusting ability of the algorithm.

In this article, the parameter $\aleph$ is set to a constant. If we associate it with iterations by letting $\aleph(t)$ be a monotonically decreasing function of t with the limit 0, then ASAMC and AESAMC will converge in distribution toward the set of global energy minima. An interesting problem is to find the decreasing rate of $\aleph(t)$ under which ASAMC and AESAMC can converge faster to the global energy minima.

5. Acknowledgments

The author's research was partially supported by grants from the National Science Foundation (DMS-0607755) and the National Cancer Institute (CA104620).

6. References

Ali, M.M., Khompatraporn, C. and Zabinsky, Z.B. (2005). A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *Journal of Global Optimization*, 31(4), 635 672.

Ali, M.M. and Storey, C. (1994). Modified controlled random search algorithms. *International Journal of Computer Mathematics*, 53(3), 229-235.

Andrieu, C., Moulines, É., and Priouret, P. (2005). Stability of Stochastic Approximation Under Verifiable Conditions. *SIAM J. Control and Optimization*, 44(1), 283-312.

Benveniste, A., Métivier, M., and Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximations*, Springer-Verlag, New York.

Bharath, B. and Borkar, V.S. (1999). Stochastic approximation algorithms: overview and recent trends. *Sadhana* 24 (4&5), 425-452.

Cappé, O., Guillin, A., Marin, J.M. and Robert, C.P. (2004). Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13(4), 907 929.

Delyon, B., Lavielle, M. and Moulines, E. (1999). Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics* 27(1), 94-128.

Dorsey, R.E. and Mayer, W.J. (1995). Genetic algorithms for estimation problems with multiple optima, non-differentiability, and other irregular features. *Journal of Business and Economic Statistics*, 13(1), 53-66.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6(6), 721-741.

Gilks, W.R., Roberts, G.O. and George, E.I. (1994). Adaptive direction sampling. *The Statistician*, 43(1), 179-189.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization, & Machine learning, Addison Wesley.

Goswami, G.R. and Liu, J.S. (2007). Evolutionary Monte Carlo methods for clustering. *Journal of Computational and Graphical Statistics*, 16(4), 855-876.

Gu, M.G. and Kong, F.H. (1998). A stochastic approximation algorithm with Markov chain Monte Carlo method for incomplete data estimation problems. *Proceedings of the National Academy of Sciences USA*, 95(13), 7270-7274.

Harth, E. and Tzanakou, E. (1974). Alopex: A stochastic method for determining visual receptive fields. *Vision Res.* 14(12), 1475-1482.

Hastings, W.K. (1970). Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 57(1), 97-109.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (2nd edition). Prentice Hall, New York.

Hoffman, D.L. and Schmidt, P. (1981). Testing the restrictions implied by the rational expectations hypothesis. *Journal of Econometrics*, 15(2), 265-287.

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

Jasra, A., Stephens, D.A. and Holmes, C.C. (2007). On population-based simulation for static inference. *Statistics and Computing*, 17(3), 263-279.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671-680.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the Institute of Electrical and Electronics Engineers*, 78(9), 1464-1480.

Liang, F. (2002). Dynamically weighted importance sampling in Monte Carlo Computation, *Journal of the American Statistical Association*, 97(459), 807-821.

Liang, F. (2005). Generalized Wang-Landau algorithm for Monte Carlo Computation. *Journal of the American Statistical Association*, 100(472), 1311-1327.

Liang, F. (2007). Annealing Stochastic Approximation Monte Carlo for Neural Network Training. *Machine Learning*, 68(3), 201-233.

Liang, F. (2008). Annealing evolutionary stochastoc approximation Monte Carlo for global optimization. Technical Report, Texas A&M University.

Liang, F., Liu, C., and Carroll, R.J. (2007). Stochastic Approximation in Monte Carlo Computation. *Journal of the American Statistical Association*, 102(477), 305-320.

Liang, F. and Wong. W.H. (2000). Evolutionary Monte Carlo sampling: applications to *Cp* model sampling and change-point problem. *Statistica Sinica*, 10(2), 317-342.

Liang, F. and Wong. W.H. (2001). Real parameter evolutionary sampling with applications in Bayesian Mixture Models, *Journal of the American Statistical Association*, 96(454), 653-666.

Mengersen, K.L. and Tweedie, R.L. (1996). Rates of convergence of the Hastings and Metropolis algorithms. *The Annals of Statistics*, 24(1), 101-121.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6), 1087- 1091.

M¨uller, P. (1991). A generic approach to posterior integration and Gibbs sampling. Technical report, Purdue University, West Lafayette, Indiana.

Price, W.L. (1983). Global optimization by controlled random search. *Journal of Optimization Theory and Applications*, 40(3), 333-348.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics* 22(3), 400-407.

Robert, C.P. and Casella, G. (2004). *Monte Carlo Statistical Methods* (2nd edition). Springer, New York.

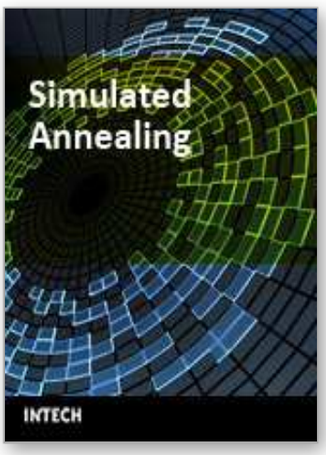
Schmitt, L.M. (2001). Theory of genetic algorithms. *Theoretical Computer Science*, 259(1), 1-61.

Spall, J.C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3), 332-341.

Storn, R. and Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4), 341-359.

Wong, W.H. and Liang, F. (1997). Dynamic weighting in Monte Carlo and optimization, *Proc. Natl. Acad. Sci. USA*, 94(26) , 14220-14224.

Zhu, H.T., Liang, F., Gu, M. and Peterson, B. (2007). Stochastic approximation algorithms for estimation of spatial mixed models. In *Handbook of Computing and Statistics with Applications*, Vol.1, S.Y. Lee (Ed.), pp.399-421, Elsevier, Amsterdam.



Simulated Annealing

Edited by Cher Ming Tan

ISBN 978-953-7619-07-7
Hard cover, 420 pages
Publisher InTech
Published online 01, September, 2008
Published in print edition September, 2008

This book provides the readers with the knowledge of Simulated Annealing and its vast applications in the various branches of engineering. We encourage readers to explore the application of Simulated Annealing in their work for the task of optimization.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Faming Liang (2008). Annealing Stochastic Approximation Monte Carlo for Global Optimization, Simulated Annealing, Cher Ming Tan (Ed.), ISBN: 978-953-7619-07-7, InTech, Available from: http://www.intechopen.com/books/simulated_annealing/annealing_stochastic_approximation_monte_carlo_for_global_optimization

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen