Stat Comput (2008) 18: 375–390
DOI 10.1007/s11222-008-9079-6

# Adaptive evolutionary Monte Carlo algorithm for optimization with applications to sensor placement problems

**Yuan Ren · Yu Ding · Faming Liang**

Received: 28 April 2008 / Accepted: 13 June 2008 / Published online: 11 July 2008

**Abstract** In this paper, we present an adaptive evolutionary Monte Carlo algorithm (AEMC), which combines a tree-based predictive model with an evolutionary Monte Carlo sampling procedure for the purpose of global optimization. Our development is motivated by sensor placement applications in engineering, which requires optimizing certain complicated "black-box" objective function. The proposed method is able to enhance the optimization efficiency and effectiveness as compared to a few alternative strategies. AEMC falls into the category of adaptive Markov chain Monte Carlo (MCMC) algorithms and is the first adaptive MCMC algorithm that simulates multiple Markov chains in parallel. A theorem about the ergodicity property of the AEMC algorithm is stated and proven. We demonstrate the advantages of the proposed method by applying it to a sensor placement problem in a manufacturing process, as well as to a standard Griewank test function.

**Keywords** Global optimization · Adaptive MCMC · Evolutionary Monte Carlo · Data mining

Y. Ren · Y. Ding
Department of Industrial and Systems Engineering,
Texas A&M University, College Station, TX 77843-3131, USA

Y. Ding
e-mail: yuding@iemail.tamu.edu

F. Liang (✉)
Department of Statistics, Texas A&M University, College Station,
TX 77843-3131, USA
e-mail: fliang@stat.tamu.edu

## 1 Introduction

Optimization problems arise in many engineering applications. Engineers often need to optimize a "black-box" objective function, i.e., a function that can only be evaluated by running a computer program. These problems are generally difficult to solve because of the complexity of the objective function and the large number of decision variables involved. Two categories of statistical methodologies, one based on random sampling and another based on predictive modeling have made great contribution to solving the optimization problems of this nature. In this article, we propose an adaptive evolutionary Monte Carlo (AEMC) method, which enhances the efficiency and effectiveness of engineering optimization problems.

A real example that motivates this research is the sensor placement problem. Simply put, in a sensor placement problem, one needs to determine the number and locations of multiple sensors so that certain design criteria can be optimized within a given budget. Sensor placement issues have been encountered in various applications, such as manufacturing quality control (Mandroli et al. 2006), structural health monitoring (Bukkapatnam et al. 2005), transportation management (Čivilis et al. 2005), and security surveillance (Brooks et al. 2003). Depending on applications, the design criteria to be optimized include, among others, sensitivity, detection probability, and coverage. A design criterion is a function of the number and locations of sensors, and this function is usually complicated and nonlinear. Evaluating the design criterion needs to run a computer program, qualifying it as a "black-box" objective function.

Mathematically, a sensor placement problem can be formulated as a constrained optimization problem.

$$\min_{w \in \mathcal{W}} H(w) \quad \text{subject to } G(w) \geq 0, \tag{1}$$

where $\mathcal{W} \subseteq \mathbf{R}^d$, $d$ is the number of sensors, $w$ is a vector of decision variables (i.e., sensor locations), $H : \mathcal{W} \to \mathbf{R}$ is a user-specified design criterion to be optimized, and $G(\cdot) \geq 0$ represents physical constraints associated with engineering systems. Taking sensor placement in an assembly process for an example, $G(\cdot) \geq 0$ means that sensors can only be installed on the surface of subassemblies, and $H(\cdot)$ is an E-optimality design criterion (Mandroli et al. 2006). In Sect. 4, we will visit this sensor placement problem with more details.

When the physical constraints are complicated and difficult to handle in an optimization routine, engineers could discretize the solution space $\mathcal{W}$ and create a finite (yet possibly huge) number of solution candidates that satisfy the constraints (see Kim and Ding 2005, Sect. 1 for an example). For the sensor placement problems, this means to identify all the viable sensor locations *a priori*; this can be done relatively easily because individual sensors are located in a low (less than or equal to three) dimensional space. One should use a high enough resolution for discretization so that "good" sensor locations are not lost. Suppose we do discretization. Then, the formulation (1) becomes an unconstrained optimization problem,

$$\min_{x \in \mathcal{X}} H(x), \tag{2}$$

where $\mathcal{X}$ is the sample space that contains the finite number of candidate sensor locations. Clearly, $\mathcal{X} \subset \mathbf{Z}^d$, which is the set of $d$-dimensional vectors with integer elements. Note that $H(\cdot)$ in (2) is still calculated according to the same design criterion as in (1) but now defined on $\mathcal{X}$. Recall that $H(\cdot)$ is of "black-box" type with potentially plenty of local optima, due to the complex nature of engineering systems.

Solving this discrete optimization problem might seem mathematically trivial because one just needs to enumerate all potential solutions exhaustively and select the best one. In most real-world applications, however, there could be an overwhelmingly large number of potential solutions to be evaluated, especially when a high-resolution discretization was performed.

Two categories of statistical methodologies exist for solving this type of optimization problems. The first category is the *sampling-based* methods: it starts with a set of random samples, and then generates new samples according to some pre-specified mechanism based on current samples and probabilistically accepts/rejects the new samples for subsequent iterations. Many well-known optimization methods, such as simulated annealing (Bertsimas and Tsitsiklis 1993), genetic algorithm (Holland 1992), and Markov chain Monte Carlo (MCMC) methods (Wong and Liang 1997), fall into this category; the differences among them come from the specific mechanism an algorithm uses to generate and accept new samples. These methods can handle complicated response surface well and have been widely applied to engineering optimizations. Their shortcoming is that they generally require a large number of function evaluations before reaching a good solution.

The second category is the *metamodel-based* methods. It also starts with a set of solution samples $\{x\}$. A metamodel is a predictive model fitted by using the historical solution pairs $\{x, H(x)\}$. With this predictive model, new solutions are generated based on the model's prediction of where one is more likely to find "good" solutions. Subsequently, the predictive model is updated as more solutions are collected. The model is labeled as "metamodel" because $H(x)$ is the computational output based on a computer model. The *metamodel-based* method originates from the research on computer experiments (Chen et al. 2006; Fang et al. 2006; Sacks et al. 1989; Simpson et al. 1997). This strategy is also called "data-mining" guided method, especially when the predictive model used therein is a classification tree model (Liu and Igusa 2007; Kim and Ding 2005; Schwabacher et al. 2001) since the tree model is a typical "data-mining" tool. For the metamodel-based or data-mining guided methods, the major shortcoming is their ineffectiveness in handling complicated response surfaces, and as a result, they only look for local optima.

This paper proposes an optimization algorithm, combining the *sampling-based* and *metamodel-based* methods. Specifically, the proposed algorithm combines evolutionary Monte Carlo (EMC) (Liang and Wong 2000, 2001) and a tree-based predictive model. The advantage of such a hybrid is that it incorporates strengths from both EMC sampling and the predictive metamodeling: the tree-based predictive model adaptively learns informative rules from past solutions so that the new solutions generated from these rules are expected to have better objective function values than the ones generated from "blind" sampling operations, while the EMC mechanism allows a search to go over the whole sample space and guides the solutions toward the global optimum. We thus label the proposed algorithm adaptive evolutionary Monte Carlo (AEMC). We will further elaborate the intuitions behind AEMC at the beginning of Sect. 3, after we review the two existing methodologies with more details in Sect. 2.

The remainder of this paper is organized as follows. Section 2 provides details of the two relevant methodologies. Section 3 describes the general idea and implementation details of the AEMC algorithm. We also prove that the AEMC algorithm preserves the ergodicity property of Markov chain samples. In Sect. 4, we employ AEMC to solve a sensor placement problem. We provide additional numerical examples to show AEMC's performance in optimization as well as its potential use for sampling. We conclude this paper in Sect. 5.

## 2 Related work

### 2.1 *Sampling-based* methods

Among the *sampling-based* methods, simulated annealing and genetic algorithms have been used to solve optimization problems for quite some time. They use different techniques to generate new random samples. Simulated annealing works by simulating a sequence of distributions determined by a temperature ladder. It draws samples according to these distributions and probabilistically accepts or rejects the samples. Geman and Geman (1984) have shown that if the temperature decreases sufficiently slowly (at a logarithmic rate), simulated annealing can reach the global optimum of $H(x)$ with probability 1. However, no one can afford such a slow cooling schedule in practice. People generally use a linearly or geometrically decreasing cooling schedule, but when doing so, the global optimum is no longer guaranteed.

Genetic algorithm uses evolutionary operators such as crossover and mutation to construct new samples. Mimicking natural selection, crossover operators are applied on two parental samples to produce an offspring that inherits characteristics of both the parents, while mutation operators are occasionally used to bring variation to the new samples. Genetic algorithm selects new samples according to their "fitness" (for example, their objective function values can be used as a measure of fitness). Genetic algorithm is known to converge to a good solution rather slowly and lacks rigorous theories to support its convergence to the global optimum.

The MCMC methods have also been used to solve optimization problems (Wong and Liang 1997; Liang 2005; Liang et al. 2007; Neal 1996). Even though a typical application of MCMC is to draw samples from complicated probability distributions, the sampling operations can be readily utilized for optimization. Consider a Boltzmann distribution $p(x) \propto \exp(-H(x)/\tau)$ for some $\tau > 0$. MCMC methods could be used to generate samples from $p(x)$. As a result, the MCMC method has a higher chance to obtain samples with lower $H(x)$ values. If we keep generating samples according to $p(x)$, we will eventually find samples close enough to the global minimum of $H(x)$. The MCMC methods perform random walks in the whole sample space and thus may potentially escape from local optima given long enough run time.

Liang and Wong (2000, 2001) proposed a method called evolutionary Monte Carlo (EMC), which incorporates many attractive features of simulated annealing and genetic algorithm into a MCMC framework. It has been shown that EMC is effective for both sampling from high-dimensional distributions and optimization problems (Liang and Wong 2000, 2001). Because EMC is an MCMC procedure, it guarantees the ergodicity of the Markov chain samples in a long run. Nonetheless, it appears that there is still a need and room to further improve the convergence rate of an EMC procedure.

Recently, adaptive proposals have been used to improve the convergence rate of traditional MCMC algorithms. For example, Gilks et al. (1998) and Brockwell and Kadane (2005) proposed to use regenerative Markov chains and update the proposal parameters at regeneration times; Haario et al. (2001) proposed an adaptive Metropolis algorithm which attempts to update the covariance matrix of the proposal distributions by making use of all past samples. Important theoretical advances on the ergodicity of the adaptive MCMC method have been made by Haario et al. (2001), Andrieu and Robert (2002), Atchadé and Rosenthal (2005), and Roberts and Rosenthal (2007).

### 2.2 *Metamodel-based* methods

In essence, *metamodel-based* methods are not much different from other sequential sampling procedures guided by a predictive model, e.g., response surface methodology used for physical experiments (Box and Wilson 1951). The metamodel based method constitutes of design and analysis of computer experiments (Chen et al. 2006; Fang et al. 2006; Sacks et al. 1989; Simpson et al. 1997) where the so-called "metamodel" is an *inexpensive* surrogate or substitute of the computer model that is oftentimes computationally *expensive* to run (e.g., the computer model could be a finite element model of a civil structure). Various statistical predictive methods have been used as the metamodels, according to the survey by Chen et al. (2006), including neural networks, tree-based methods, Splines, and spatial correlation models. During the past few years, there have emerged a number of research developments, labeled as data-mining guided engineering designs (Guikema et al. 2004; Huyet 2006; Liu and Igusa 2007; Kim and Ding 2005; Michalski 2000; Schwabacher et al. 2001). The data-mining guided methods are basically one form of metamodel-based methods because they also use a statistical predictive model to guide the selection of design solutions. The predictive models used in the data-mining guided designs include regression, classification tree, and clustering methods.

When looking for an optimal solution, the predictive model is used as follows. After fitting a metamodel (or simply a model, in the cases of physical experiments), one could use it to predict where good solutions are more likely to be found and thus select subsequent samples accordingly. This sampling-modeling-prediction procedure is considered a data-mining operation. Liu and Igusa (2007) and Kim and Ding (2005) demonstrated that the data-mining operation could greatly speed up computation under right circumstances. Compared with the slow converging *sampling-based* methods, the *metamodel-based* methods can be especially useful when one has limited amount of data samples; this happens when physical experiments or computer simulations are expensive to conduct. But the metamodel based methods are "greedy" search methods and can be easily entrapped in local optima.

## 3 Adaptive evolutionary Monte Carlo

### 3.1 General idea of AEMC

The strengths as well as the limitations of the *sampling-based* and the *metamodel-based* search methods motivate us to combine the two schemes and develop the AEMC algorithm. The intuition behind how AEMC works is explained as follows.

A critical shortcoming of the *metamodel-based* methods is that their effectiveness highly depends on how representative the sampled solutions are of the optimal regions of the sample space. Without representative data, the resulting metamodel could mislead the search to non-optimal regions. Consequently, the subsequent sampling from those regions will not help the search get out of the trap. In particular, when the sample space is large and good solutions only lie in a small portion of the space, data obtained by a uniform sampling from the sample space will not be representative enough. Taking the sensor placement problem shown later in this paper for an example, we found that only 5% of the solutions have relatively good objective function values. Under this circumstance, stand-alone metamodeling mechanism could hardly be effective (as shown in the numerical results in Sect. 4), thereby promoting the need to improve the sample quality for the purpose of establishing a better metamodel.

It turns out that *sampling-based* algorithms (we choose EMC in this paper), though slow as a stand-alone optimization tool, are able to improve the quality of the sampled solutions. This is because when conducting random searches over a sample space, EMC will gradually converge in distribution to the Boltzmann distribution in (4), i.e., the smaller the value of $H(x)$ is, the higher the probability of sampling $x$ is (recall that we want to minimize $H(x)$). In other words, EMC will iteratively and stochastically direct current samples toward the optimal regions such that the visited solutions are more representative of the optimal regions of the sample space. With the representative samples produced by EMC, a metamodeling operation could generate more accurate predictive models to characterize the promising subregions of the sample space.

The primary tool for improving the *sampling-based* search is to speed up its converge rate. As argued in Sect. 2, making a MCMC method adaptive is an effective way of achieving such an objective. The metamodel part of AEMC learns the function surface of $H(x)$, and allows us to construct more effective proposal distributions for subsequent sampling operations. As argued in Gilks et al. (1995), the rate of convergence of a Markov chain to the Boltzmann distribution in (4) depends crucially on the relationship between the proposal function and the target function $H(x)$. To the best of our knowledge, AEMC is also the first adaptive MCMC method that simulates multiple Markov chains in parallel, while the existing adaptive MCMC methods are all based on simulation of one single Markov chain. So the AEMC can utilize information from multiple chains to improve the convergence rate.

The above discussions explain the benefit of combining the *metamodel-based* and *sampling-based* method and executing them alternately in a fashion shown in Fig. 1.

In the sequel, we will present the details of the proposed AEMC algorithm. For metamodeling (or data-mining) operations, we use classification and regression trees (CART), proposed by Breiman et al. (1984), to fit predictive models. We choose CART primarily because of its computational efficiency. Our goal of solving an optimization problem requires the data-mining operations to be fast and computationally scalable in order to accommodate large-sized data sets. Since the data-mining operations are repeatedly used, a complicated and computationally expensive method will unavoidably slow down the optimization process.

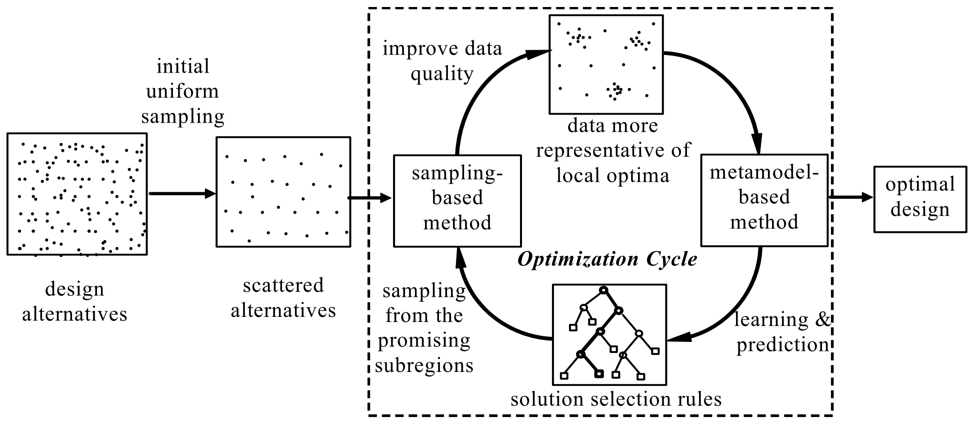

**Fig. 1** General framework of combining *sampling-based* and *metamodel-based* methods

## 3.2 Evolutionary Monte Carlo

For the convenience of reading this paper, we provide a brief summary of EMC in this section and a description of the operators of EMC in Appendix A. Please refer to Liang and Wong (2000, 2001) for more details. EMC integrates features of simulated annealing and genetic algorithm into a MCMC framework. Similar to simulated annealing, EMC uses a temperature ladder and simultaneously simulates a population of Markov chains, each of which is associated with a different temperature. The chains with high temperatures can easily escape from local optima, while the chains with low temperatures can search around some local regions and find better solutions faster. The population is updated by crossover and mutation operators, just like genetic algorithm, and therefore adopts some level of "learning" capability, i.e., samples with better fitness will have a greater probability of being selected and pass their good "genetic materials" to the offsprings.

A *population*, as mentioned above, is actually a set of $n$ solution samples. The state space associated with a population is the product of $n$ sample spaces, namely $\mathcal{X}^n = \mathcal{X} \times \cdots \times \mathcal{X}$. Denote a population $\boldsymbol{x} \in \mathcal{X}^n$ such that $\boldsymbol{x} = \{x_1, \ldots, x_n\}$, where $x_i = \{x_{i1}, \ldots, x_{id}\} \in \mathcal{X}$ is the $i$-th $d$-dimensional solution sample. EMC attaches a different temperature, $t_i$, to a sample $x_i$, and the temperatures form a ladder with the ordering $t_1 \geq \cdots \geq t_n$. We denote $\boldsymbol{t} = \{t_1, \ldots, t_n\}$. Then the Boltzmann density can be defined for a sample $x_i$ as

$$f_i(x_i) = \frac{1}{Z(t_i)} \exp\{-H(x_i)/t_i\}, \tag{3}$$

where $Z(t_i)$ is a normalizing constant, and

$$Z(t_i) = \sum_{\{x_i\}} \exp\{-H(x_i)/t_i\}.$$

Assuming that samples in a population are mutually independent, we then have the Boltzmann distribution of the population as

$$f(\boldsymbol{x}) = \prod_{i=1}^{n} f_i(x_i) = \frac{1}{Z(\boldsymbol{t})} \exp\left\{-\sum_{i=1}^{n} H(x_i)/t_i\right\}, \tag{4}$$

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

Given an initial population $\boldsymbol{x}^{(0)} = \{x_1^{(0)}, \ldots, x_n^{(0)}\}$ and the temperature ladder $\boldsymbol{t} = \{t_1, \ldots, t_n\}$, $n$ Markov chains are simulated simultaneously. Denote the iteration index by $k = 1, 2, \ldots$, and the $k$-th iteration of EMC consists of two steps:

1. With probability $p_m$ (mutation rate), apply a mutation operator to each sample independently in the population $\boldsymbol{x}^{(k)}$. With probability $1 - p_m$, apply a crossover operator to the population $\boldsymbol{x}^{(k)}$. Accept the new population according to the Metropolis-Hastings rule. Details are given in Appendices A.1 and A.2.
2. Try to exchange $n$ pairs of samples $(x_i^{(k)}, x_j^{(k)})$, with $i$ uniformly chosen from $\{1, \ldots, n\}$ and $j = i \pm 1$ with probability $w(x_j^{(k)} | x_i^{(k)})$ as described in Appendix A.3.

EMC is a standard MCMC algorithm and thus maintains the ergodicity property for its Markov chains. Because of incorporation of features from simulated annealing and genetic algorithm, EMC constructs proposal distributions more effectively and converges faster than traditional MCMC algorithms.

## 3.3 The AEMC algorithm

In AEMC, we first run a number of iterations of EMC and then use CART to learn a proposal distribution (for generating new samples) from the samples produced by EMC. Denote by $\boldsymbol{\Theta}^{(k)}$ the set of samples we have retained after iteration $k$. From $\boldsymbol{\Theta}^{(k)}$, we define high performance samples to be those with relatively small $H(x)$ values. The high performance samples are the representatives of the promising search regions. We denote by $H_{(h)}^{(k)}$ the $h$ percentile of the $H(x)$ values in $\boldsymbol{\Theta}^{(k)}$. Then, the set of high performance samples at iteration $k$, $\boldsymbol{H}^{(k)}$, are defined as

$$\boldsymbol{H}^{(k)} = \{x : x \in \boldsymbol{\Theta}^{(k)} \text{ and } H(x) \leq H_{(h)}^{(k)}\}.$$

As a result, the samples in $\boldsymbol{\Theta}^{(k)}$ are grouped into two classes, the high performance samples in $\boldsymbol{H}^{(k)}$ and the others. Treating these samples as a training dataset, we then fit a CART model to a two-class classification problem. Using the prediction from the resulting CART model, we can partition the sample space into rectangular regions, some of which have small $H(x)$ values and are therefore deemed as the promising regions, while other regions as non-promising regions.

The promising regions produced by CART are represented as $a_j^{(k)} \leq x_{ij} \leq b_j^{(k)}$, $j = 1, \ldots, d$, $i = 1, \ldots, n$. Since $\mathcal{X}$ is discrete and finite, there is a lower bound $l_j$ and an upper bound $u_j$ in the $j$-th dimension of the sample space. Clearly we have $l_j \leq a_j^{(k)} \leq b_j^{(k)} \leq u_j$. CART may produce multiple promising regions. We denote by $m^{(k)}$ the number of regions. Then, the collection of the promising regions is specified in the following:

$$a_{js}^{(k)} \leq x_{ij} \leq b_{js}^{(k)},$$
$$j = 1, \ldots, d, \; i = 1, \ldots, n, \; s = 1, \ldots, m^{(k)}. \tag{5}$$

As the algorithm goes on, we continuously update $\boldsymbol{\Theta}^{(k)}$, and hence $a_{js}^{(k)}$ and $b_{js}^{(k)}$.

After we have identified the promising regions, the proposal density is constructed based on the following thoughts:

get a sample from the promising regions with probability $R$, and from elsewhere with probability $1-R$, respectively. We recommend using a relatively large $R$ value, say $R=.9$. Since there may be multiple promising regions identified by CART, we denote the proposal density associated with each region by $q_{ks}(x)$, $s=1,\dots,m^{(k)}$. In this paper, we use a Metropolis-within-Gibbs procedure (Müller 1991) to generate new samples as follows.

For $i=1,\dots,n$, denote the population after the $k$-th iteration by $\boldsymbol{x}^{(k+1,i-1)}=(x_1^{(k+1)},\dots,x_{i-1}^{(k+1)},x_i^{(k)},\dots,x_n^{(k)})$, of which the first $i-1$ samples have been updated, and the Metropolis-within-Gibbs procedure is about to generate the $i$-th new sample. Note that $\boldsymbol{x}^{(k+1,0)}=(x_1^{(k)},\dots,x_n^{(k)})$.

1. Set $S$ to be randomly chosen from $\{1,\dots,m^{(k)}\}$. Generate a sample $x_i'$ from the proposal density $q_{kS}(\cdot)$.

$$q_{kS}(x_i')=\prod_{j=1}^{d}\Bigg(r\frac{I(a_{jS}^{(k)}\le x_{ij}'\le b_{jS}^{(k)})}{b_{jS}^{(k)}-a_{jS}^{(k)}} + (1-r)\frac{I(x_{ij}'<a_{jS}^{(k)}\text{ or }x_{ij}'>b_{jS}^{(k)})}{(u_j-l_j)-(b_{jS}^{(k)}-a_{jS}^{(k)})}\Bigg), \qquad (6)$$

where $I(\cdot)$ is the indicator function. Here $r$ is the probability of sampling uniformly within the range specified by the CART rules on each dimension. Since each dimension is independent of each other, we have $R=r^d$.

2. Construct a new population $\boldsymbol{x}^{(k+1,i)}$ by replacing $x_i^{(k)}$ with $x_i'$, and accept the new population with probability $\min(1,r_d)$, where

$$r_d=\frac{f(\boldsymbol{x}^{(k+1,i)})}{f(\boldsymbol{x}^{(k+1,i-1)})}\frac{T(\boldsymbol{x}^{(k+1,i-1)}|\boldsymbol{x}^{(k+1,i)})}{T(\boldsymbol{x}^{(k+1,i)}|\boldsymbol{x}^{(k+1,i-1)})} = \exp\{-(H(x_i')-H(x_i^{(k)}))/t_i\} \times\frac{T(\boldsymbol{x}^{(k+1,i-1)}|\boldsymbol{x}^{(k+1,i)})}{T(\boldsymbol{x}^{(k+1,i)}|\boldsymbol{x}^{(k+1,i-1)})}, \qquad (7)$$

If the proposal is rejected, $\boldsymbol{x}^{(k+1,i)}=\boldsymbol{x}^{(k+1,i-1)}$.

The transition probability in (7) is calculated as follows. Since we only change one sample in each Metropolis-within-Gibbs step, the transition probability can be written as $T(x_i^{(k)}\to x_i'|\boldsymbol{x}_{[-i]}^{(k)})$, where $\boldsymbol{x}_{[-i]}^{(k)}=(x_1^{(k+1)},\dots,x_{i-1}^{(k+1)},x_{i+1}^{(k)},\dots,x_n^{(k)})$. Then we have

$$T(x_i^{(k)}\to x_i'|\boldsymbol{x}_{[-i]}^{(k)})=\sum_{s=1}^{m^{(k)}}\frac{1}{m^{(k)}}q_{ks}(x_i').$$

From (6), it is not difficult to see that as long as $0<r<1$, the proposal is global, i.e., $q_{ks}(x)>0$ for all $x\in\mathcal{X}$. Since $\mathcal{X}$ is finite, it is natural to assume that $f(x)$ is bounded away from 0 and $\infty$ on $\mathcal{X}$. Thus, the minorisation condition (Mengersen and Tweedie 1996), i.e.,

$$\omega^*=\sup_{x\in\mathcal{X}}\frac{f(x)}{q_{ks}(x)}<\infty,$$

is satisfied. As shown in Appendix C, satisfaction of this condition would lead to the ergodicity of the AEMC algorithm.

Now we are ready to present a summary of the AEMC algorithm, which consists of two modes: the EMC mode and the data-mining (or metamodeling) mode.

1. Set $k=0$. Start with an initial population $\boldsymbol{x}^{(0)}$ by uniformly sampling $n$ samples over $\mathcal{X}$ and a temperature ladder $\boldsymbol{t}=\{t_1,\dots,t_n\}$.
2. EMC mode: run EMC until a switching condition is met.
   - Apply mutation, crossover, and exchange operators to the population $\boldsymbol{x}^{(k)}$ and accept the updated population according to the Metropolis-Hastings rule. Set $k=k+1$.
3. Run the data-mining mode until a switching condition is met.
   - With probability $P_k$, use the CART method to update the promising regions, i.e., update the values of $a_{js}^{(k+1)}$ and $b_{js}^{(k+1)}$ in (5).
   - With probability $1-P_k$, do not apply CART and simply let $a_{js}^{(k+1)}=a_{js}^{(k)}$ and $b_{js}^{(k+1)}=b_{js}^{(k)}$.
   - Generate $n$ new samples following the Metropolis-within-Gibbs procedure mentioned earlier in this section. Set $k=k+1$.
4. Alternate between the two modes until a stopping rule is met. The algorithm could terminate when the computational budget (the number of iterations) is consumed or when the change in the best $H(x)$ value does not exceed a given threshold for several iterations.

To effectively implement AEMC, several issues need to be considered. Firstly, it is the choice of parameters in EMC: $n$ and $p_m$. We simply follow the recommendations made in the EMC related research. So we set $n$ and $p_m$ to values that favor EMC. Typically, $n=5$–$20$ and $p_m\approx.25$ (Liang and Wong 2000).

Secondly, the choice of $P_k$. We need to make sure $P_1>P_2>\cdots>P_k>\cdots$, and $\lim_{k\to\infty}P_k=0$, which ensures the diminishing adaptation condition required for the ergodicity of the adaptive MCMC algorithms (Roberts and Rosenthal 2007). As discussed in Appendix C, meeting the diminishing adaptation condition is crucial to the convergence of AEMC. Intuitively, $P_k$ could be considered as the "learning rate". In the beginning of the algorithm, we do not have much information about the function surface, and therefore we apply the data-mining mode to learn new information with a relatively high probability. As the algorithm goes on, we may have sufficient knowledge about the function surface, and it may be a waste to execute the data-mining mode too often. So we make the "learning rate" decrease over time. Specifically, we set $P_k = 1/k^\delta$. The $\delta$ ($\delta > 0$) controls the decreasing speed of $P_k$. The larger $\delta$ is, the faster $P_k$ decreases to 0. We choose $\delta = .1$ in this paper.

Thirdly, the construction of training samples $\boldsymbol{\Theta}^{(k)}$. The question is that should we use all the past samples to construct $\boldsymbol{\Theta}^{(k)}$ or should we use a subset instead, for example, using only recent samples gathered since the last data-mining operation. If we use all the past samples, data mining will be performed on a large dataset, and doing so will inevitably take a long time and thus slow down the optimization process. Because EMC is able to randomly sample from the whole sample space, AEMC is less likely to fall into local optima even if we just use recent samples. Thus, the latter becomes our choice.

Lastly, we discuss the following tuning parameters of AEMC.

– Switching condition $M$. In order to adopt the strengths of the two mechanisms and compensate their weaknesses, a proper switching condition is needed to select the appropriate mode of operations for the proposed optimization procedure. Because of the inability of a stand-alone data-mining mode to find representative samples (will be shown in Sect. 4), it is not beneficial to run the data-mining mode for multiple iterations. So a natural choice is to run the data-mining mode only once for every $M$ iterations of EMC. If $M$ is too large, the learning effect from the data-mining mode will be overshadowed and AEMC virtually becomes a pure EMC algorithm. If $M$ is too small, EMC may not be able to gather enough representative data and thus data mining could hardly be effective. We recommend choosing $M$ based on the value of $n \times M$, which is the same size used in the data-mining mode for establishing the predictive model. From our experience, $nM = 300$–$500$ works quite well.
– The proper choice of $h$ varies for different problems. For a minimization problem, a large $h$ value could bring many uninteresting samples into the set of supposedly high performance solutions and then slow down the optimization process. On the other hand, a small value of $h$ would increase the chance for the algorithm to fall into local optimum. Besides, a very small $h$ value may lead to small promising regions, which could make the acceptance rate of new samples too low. But the danger of falling into the local optima is not grave because the data-mining mode is followed by EMC that randomizes the population again and makes it possible to escape from the local optima. In light of this, we recommend an aggressive choice for the $h$ value, i.e. $h = 5$–$15\%$.
– Choice of the tree size in CART. Fitting a tree is to approximate the response surface of $H(x)$. A small-sized tree may not be sophisticated enough to approximate the surface, while a large-sized tree may overfit the data. Since we apply CART multiple times in the entire procedure of AEMC, we believe that the mechanism of how the trees work in AEMC is actually similar to tree boosting (Hastie et al. 2001), where a series of CART are put together to produce a result. For tree boosting, Hastie et al. (2001) recommended $2 \le J \le 10$.

  In our problem, however, controlling $J$ alone does not precisely fulfill our objective. Because our goal is to find the global optimum rather than to make good prediction for the whole response surface, we are much more interested in the part of the response surface where the $H(x)$ values are relatively small, corresponding to the class of high performance samples, $\boldsymbol{H}^{(k)}$. It then makes sense to control the number of terminal nodes associated with $\boldsymbol{H}^{(k)}$, denoted by $J_H$. Controlling $J_H$ enables us to fit a CART model that better approximates a high performance portion of the response surface. Note that the set of terminal nodes representing $\boldsymbol{H}^{(k)}$ is a subset of all terminal nodes in the corresponding tree, meaning that the value of $J_H$ is positively correlated with the value of $J$. Thus, controlling $J_H$ in the meanwhile also regulates the value of $J$. The basic rationale behind the selection of $J_H$ is similar to that for $J$: a large $J_H$ results in a large $J$ and could lead to overfitting; the danger of using too small a $J_H$ is that there will be too few promising regions for the subsequent actions to search and evolve from, which may cause the proposed procedure to miss some good samples. From the above arguments, we note that the $\boldsymbol{H}^{(k)}$ in our problem plays an analogous role as the whole response surface in the traditional tree boosting. We believe that the guideline for $J$ could be transferred to $J_H$, i.e., $2 \le J_H \le 10$.

In Sect. 4, we shall provide a sensitivity analysis of the three tuning parameters, which reveals how the performance of AEMC depends on their choices. From the results, we will see that $M$ and $h$ are the two most important tuning parameters and that AEMC is not sensitive to the value of $J_H$, provided that it is chosen from the above-recommended range.

As an adaptive MCMC algorithm, AEMC simulates multiple Markov chains in parallel and could therefore utilize the information from different chains for improving the convergence rate. We will provide some empirical results in support of this claim in Sect. 4 because a theoretical assessment of convergence rate is too difficult to obtain. But we do investigate the ergodicity of AEMC. We are able to show that AEMC is ergodic as long as the proposal $q_{ks}(\cdot)$ is global and the data-mining mode is run with probability $P_k \to 0$ as $k \to \infty$. The ergodicity implies that AEMC will reach the global optimum of $H(x)$ given enough run time. This property of the AEMC algorithm is stated in Theorem 1 and its proof is included in Appendix C.

**Theorem 1** *If* $0 < r < 1$, $\lim_{k\to\infty} P_k = 0$, *and* $\mathcal{X}$ *is compact, then AEMC is ergodic, i.e., the samples* $\boldsymbol{x}^{(k)}$ *converge in distribution to* $f(\boldsymbol{x})$.

A final note is that we assume the sample space $\mathcal{X}$ to be discrete and bounded in this paper. Yet the AEMC algorithm could easily be extended to an optimization problem with a continuous and bounded sample space. One just needs to use the mutation and crossover operators that are proposed in Liang and Wong (2001). Theorem 1 still holds. For unbounded sample spaces, some constraints can be put on the tails of the distribution $f(\boldsymbol{x})$ and the proposal distribution to ensure that the minorisation condition hold. Refer to Roberts and Tweedie (1996), Rosenthal (1995) and Roberts and Rosenthal (2004) for more discussions on this issue.

## 4 Numerical results

To illustrate the effectiveness of the AEMC algorithm, we use it to solve three problems: the first two are for optimization purposes and the third is for sampling purposes. The first example is a sensor placement problem in an assembly process, and in the second example we optimize the Griewank function (Griewank 1981), a widely used test function in the area of global optimization. In the third example we use AEMC to sample from a mixture Gaussian distribution and see how AMEC, as an adaptive MCMC method, can help the sampling process.

For the two optimization examples, we compare AEMC with EMC, the stand-alone CART guided method, and the standard genetic algorithm. As to the parameters in AEMC, we chose $n = 5$, $p_m = .25$, $M = 60$, $J_H = 6$ and $h = 10\%$. For the standard genetic algorithm, we let the population size be 100, crossover rate be .9 and mutation rate be .01. All optimization algorithms were implemented in the MATLAB environment, and all reported performance statistics of the algorithms were the average result of 10 trials. The performance indices for comparison include the best function value found by an algorithm and the number of times that $H(\cdot)$ has been evaluated (also called "the number of function evaluations" hereinafter). The use of the number of function evaluations as a performance measure makes good sense for many engineering design problems, where the objective function $H(\cdot)$ is complex and time consuming to evaluate. Thus the time of function evaluations essentially dominates the entire computational cost. In Sect. 4.3, we will also present a sensitivity analysis on the three tuning parameters, the switching condition $M$, the percentile value $h$, and the tree size $J_H$.

### 4.1 Sensor placement example

In this section, we attempt to find an optimal sensor placement strategy in a three-station two-dimensional (2-D) assembly process (Fig. 2). Coordinate sensors are distributed throughout the assembly process to monitor the dimensional quality of the final assembly and/or of the intermediate subassemblies. $M_1$–$M_5$ are five coordinate sensors that are currently in place on the three stations; this is simply one instance of, out of hundreds of thousands of other possible, sensor placements.

The goal of having these coordinate sensors is to estimate the dimensional deviation at the fixture locators on different stations, labeled as $P_i$, $i = 1, \ldots, 8$, in Fig. 2. Researchers have established physical models connecting the sensor measurements to the deviations associated with the fixture locators (Jin and Shi 1999; Mandroli et al. 2006). Such a relationship could be expressed in a linear model, mathematically equivalent to a linear regression model. Thus, the design of sensor placement becomes very similar to the optimal design problem in experimentation, and the problem to

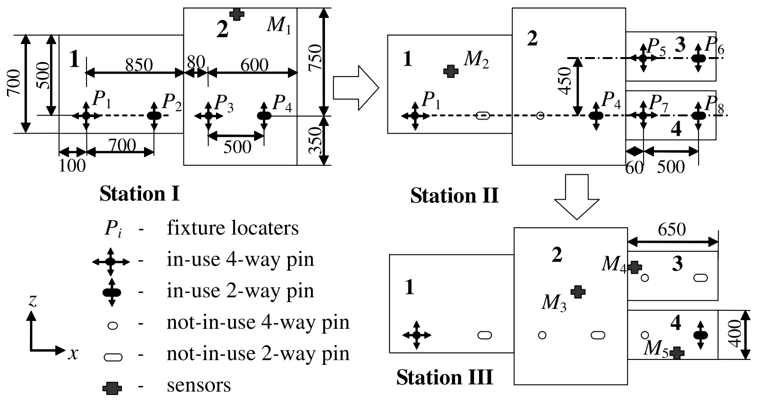

**Fig. 2** Illustration: a multi-station assembly process. The process proceeds as follows: (i) at the station I, part 1 and part 2 are assembled; (ii) at the station II, the subassembly consisting of part 1 and part 2 receives part 3 and part 4; and (iii) at the station III, no assembly operation is performed but the final assembly is inspected. The 4-way pins constrain the part motion in both the $x$- and the $z$-axes, and the 2-way pins constrain the part motion in the $z$-axis

decide where one should place them on respective assembly stations so that the estimation of the parameters (i.e., fixturing deviations) can be achieved with the minimum variance. Similar to the optimal experimental designs, people chose to optimize an alphabetic optimality criterion (such as D-optimality or E-optimality) of an information matrix that is determined by the corresponding sensor placement. In this paper, we use the E-optimality design criterion as a measure of the sensor system sensitivity, the same as in Liu et al. (2005). But the AMEC algorithm is certainly applicable to other design criteria. Due to the complexity of this sensor placement problem, one will need to run a set of MATLAB codes to calculate the response of sensitivity for a given placement of sensors. For more details of the physical process and modeling, please refer to Mandroli et al. (2006).

We want to maximize the sensitivity, which is equivalent to minimizing the maximum variance of the parameter estimation. To facilitate the application of the AEMC algorithm, we discretize the geometric area of each part viable for sensor placement using a resolution of 10 mm (which is the size of a locator's diameter); this treatment is the same as what was done in Kim and Ding (2005) and Liu et al. (2005). The discretization procedure also ensures that all constraints are incorporated into the candidate samples so that we can solve the unconstrained optimization (2) for sensor placement. This discretization results in $N_c$ candidate locations for any single sensor so the sample space for (2) is $\mathcal{X} = [1, N_c]^d \cap \mathbf{Z}^d$. For the assembly process shown in Fig. 2, the 10-mm resolution level results in the number of candidate sensor locations on each part as $n_1 = 6{,}650$, $n_2 = 7{,}480$, $n_3 = 2{,}600$, and $n_4 = 2{,}600$. Because part 1 and 2 appear on all three stations, part 3 and 4 appear on the second and third stations, there are totally $N_c = 3 \times (n_1 + n_2) + 2 \times (n_3 + n_4) = 52{,}790$ candidate locations for each sensor. Suppose that $d = 9$, meaning that nine sensors are to be installed, then the total number of solution candidates is $C_9^{52{,}790} \approx 8.8 \times 10^{36}$, where $C_a^b$ is the combinational operator. Evidently, the number of solution candidates is overwhelmingly large.

Moreover, we want to *maximize* the sensitivity objective function (i.e., more sensitive a sensor system, the better), while AEMC is to solve a minimization problem. For this reason, we let $H(x)$ in the AEMC algorithm equal to the sensitivity response of $x$ multiplied by $-1$, where $x$ represents an instance of sensor placement. We solve the optimal sensor placement problem for nine sensors and 20 sensors, respectively.

For the scenario of $d = 9$, each algorithm was run for $10^5$ function evaluations. The results of various methods are presented in Fig. 3. It demonstrates a clear advantage of AEMC over the other algorithms. EMC and genetic algorithm have similar performances. After about $4 \times 10^4$ function evaluations, AEMC finds $H(x) \approx 1.20$, which is the best value found by EMC and genetic algorithm after $10^5$ function evaluations. This translates to 2.5 times improvement in terms of CPU time. Figure 4 gives a Boxplot of the best sensitivity values found at the end of each algorithm. AMEC finds a sensitivity value, on average, 10% better than EMC and genetic algorithm. AEMC also has smaller uncertainty than EMC. From the two figures, it is worth noting that the stand-alone CART guided method performs much worse than the other algorithms in this example. We believe this happens mainly because the stand-alone CART guided method fails to gather representative data in the sample space associated with the problem. Figure 5 presents the best (i.e., yielding the largest sensitivity) sensor placement strategy found in this example.

We also test the AEMC method in a higher dimensional case, i.e., when $d = 20$. All the algorithms were again run for $10^5$ function evaluations. The algorithm performance curves are presented in Fig. 6, where we observe that the sensitivity value found by AEMC after $3 \times 10^4$ function evaluations is the same as that found by EMC after $10^5$ function evaluations. This translates to a 3-fold improve-

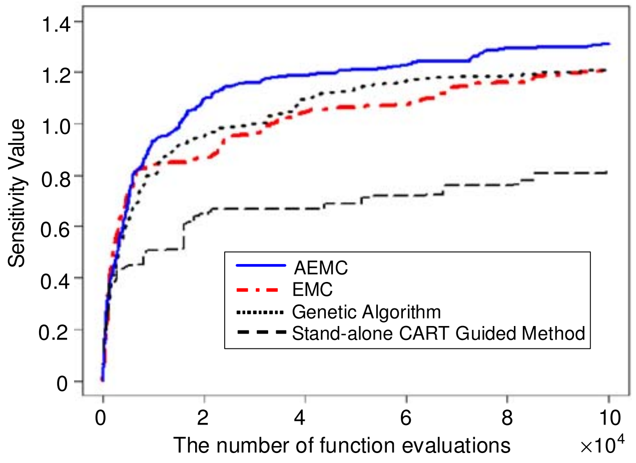

**Fig. 3** Performances of the various algorithms for nine sensors

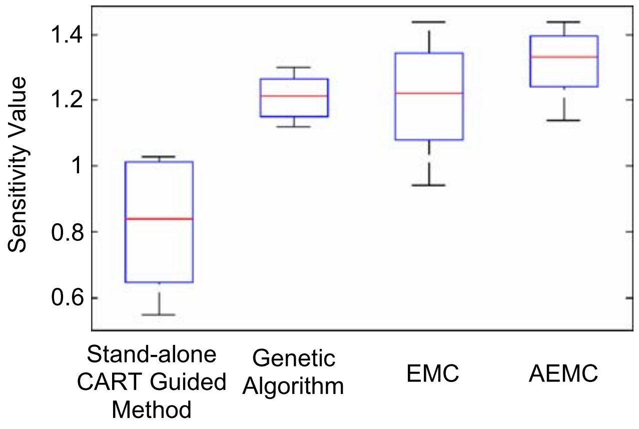

**Fig. 4** Uncertainty of different algorithms for nine sensors

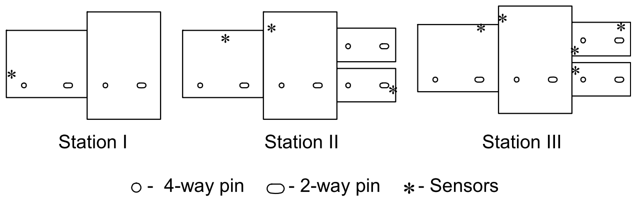

**Fig. 5** Best sensor placement for nine sensors

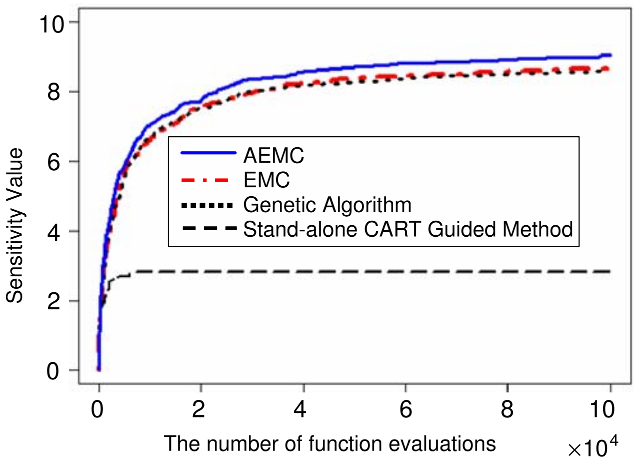

**Fig. 6** Performances of the various algorithms for 20 sensors

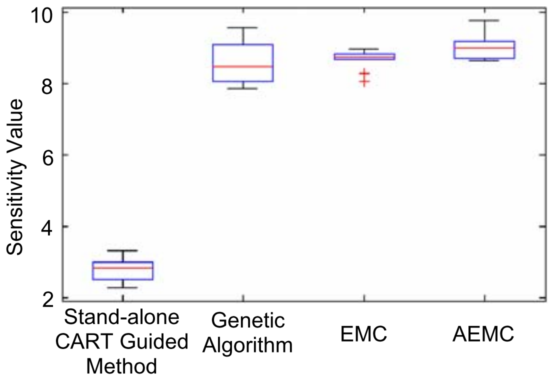

**Fig. 7** Uncertainty of different algorithms for 20 sensors

ment in terms of CPU time. Interestingly, this improvement is greater than that in the 9-sensor case. The final sensitivity value attained by AMEC is, on average, 7% better than EMC and genetic algorithm. Again, the stand-alone CART guided method fails to compete with the other algorithms. We feel that as the dimensionality of the sample space gets higher, the performance of the stand-alone CART guided method gets worse compared to others. Figure 7 shows the uncertainty of each algorithm. In this case, AEMC has a little higher uncertainty than EMC, but the average results of AEMC are still better. Figure 8 presents the best sensor placement strategy found in this example.

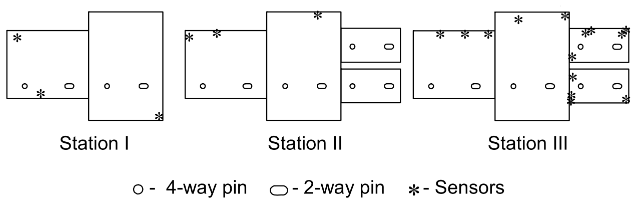

**Fig. 8** Best sensor placement for 20 sensors

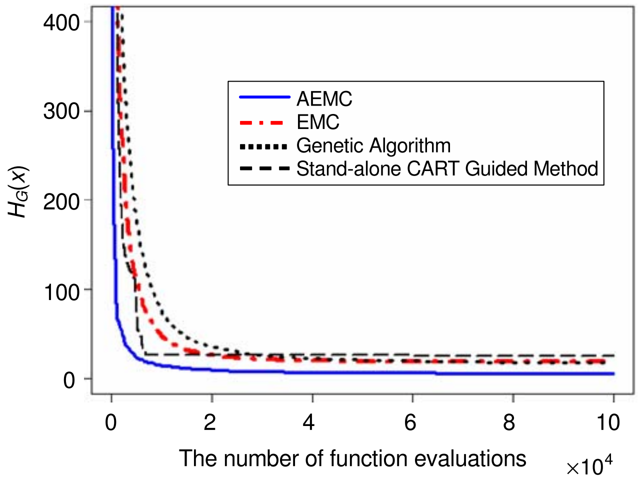

**Fig. 9** Performances of the various algorithms for the Griewank function

## 4.2 Griewank test function

In order to show the potential applicability of AEMC to other optimization problems, we test it on a well-known test function. The Griewank function (Griewank 1981) has been used as a test function for global optimization algorithms in a broad body of literature. The function is defined as

$$H_G(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$$
$$-600 \le x_i \le 600, \; i = 1, \dots, d.$$

The global minimum is located at the origin and the global minimum value is 0. The function has a very large number of local minima, exponentially increasing with $d$. Here we set $d = 50$.

Figure 9 presents the performances of the various algorithms. All algorithms were run for $10^5$ function evaluations. Please note that we have truncated the $y$-axis to be between 0 and 400 so as to show the performances at the early-stage of different algorithms more clearly. AEMC clearly outperforms the other algorithms, especially in the beginning stage, as one can observe that AEMC converges much faster. As explained earlier, this fast convergence is an appealing property to engineering design problems. In this example,

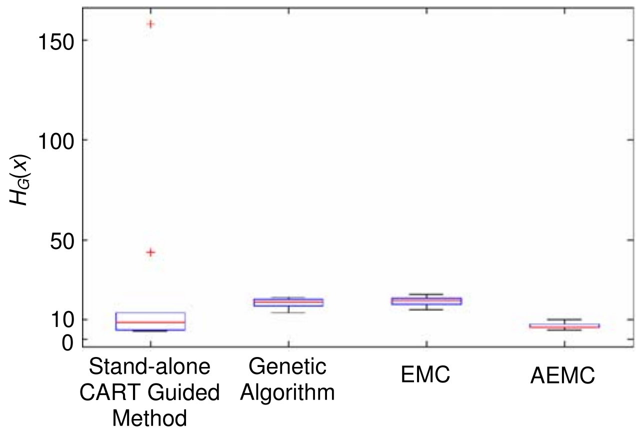

**Fig. 10** Uncertainty of different algorithms for the Griewank function

the stand-alone CART guided method converges faster than genetic algorithm and EMC, but is entrapped into a local optimum at an early stage. AEMC appears to level off after 65,000 function evaluations. However, as assured by Theorem 1, AEMC will eventually reach the global optimum if given enough computational effort. Figure 10 gives a Boxplot of the best $H_G(x)$ found at the end of the algorithms. Comparing to the other methods, the AEMC algorithm not only improves the average performance but also reduces the uncertainty. Although the stand-alone CART guided method has found good solutions, the uncertainty of this algorithm is much higher than AEMC and the other two.

## 4.3 Sensitivity analysis

We run an ANOVA analysis to investigate how sensitive the performance of AEMC to the tuning parameters: the switching condition $M$, the percentile value $h$, and the tree size $J_H$. The value of $M$ is chosen from five levels (10, 30, 60, 90, 120), the $h$ is chosen from three levels (1%, 10%, 20%), and the $J_H$ is chosen from three levels (3, 6, 12). Then a full factorial design with 45 cases is constructed.

For the sensor placement example, we use the 9-sensor case for the ANOVA analysis. AEMC was run for $10^5$ function evaluations, and we recorded the best function value found as the output. For each factor level combination, this was done five times. The ANOVA table is shown in Table 1. We can see that the main effect of $M$ and $h$ is significant at the .05 level. Our study also revealed that relatively smaller $M$ and $h$ are favored. For the Griewank example, we ran AEMC for $5 \times 10^4$ function evaluations and recorded the best function value found as the output. For each factor level combination, this was done five times. The ANOVA table is shown in Table 2. If using .05 level, then we can see that the main effects of $M$ and $h$ are significant. Here smaller $h$ and $M$ are favored as well.

**Table 1** ANOVA analysis for the sensor placement example

| Source | Sum sq. | D.f. | Mean sq. | $F$ | Prob > $F$ |
|---|---|---|---|---|---|
| $M$ | 0.33 | 4 | 0.08 | 2.43 | 0.05 |
| $h$ | 0.87 | 2 | 0.44 | 12.75 | 0.00 |
| $J_H$ | 0.06 | 2 | 0.03 | 0.84 | 0.43 |
| $M \times h$ | 0.26 | 8 | 0.03 | 0.94 | 0.48 |
| $M \times J_H$ | 0.14 | 8 | 0.02 | 0.53 | 0.84 |
| $h \times J_H$ | 0.11 | 4 | 0.03 | 0.79 | 0.53 |
| Error | 6.70 | 196 | 0.03 | | |
| Total | 8.47 | 224 | | | |

**Table 2** ANOVA analysis for the Griewank example

| Source | Sum sq. | D.f. | Mean sq. | $F$ | Prob > $F$ |
|---|---|---|---|---|---|
| $M$ | 2393.40 | 4 | 598.35 | 48.50 | 0.00 |
| $h$ | 1046.26 | 2 | 523.13 | 42.40 | 0.00 |
| $J_H$ | 0.18 | 2 | 0.09 | 0.01 | 0.99 |
| $M \times h$ | 163.51 | 8 | 20.44 | 1.66 | 0.11 |
| $M \times J_H$ | 104.15 | 8 | 13.02 | 1.06 | 0.40 |
| $h \times J_H$ | 43.41 | 4 | 10.85 | 0.88 | 0.48 |
| Error | 2418.08 | 196 | 12.34 | | |
| Total | 6169.00 | 224 | | | |

Based on our sensitivity analysis for both examples, we understand that the switching condition $M$ and the percentile value $h$ are the important factors affecting the performance of AEMC. To choose suitable values for $M$ and $h$, users may follow our general guidelines outlined in Sect. 3.3 and further tune their values for specific problems.

## 4.4 Sampling from a mixture Gaussian distribution

AEMC falls into the category of adaptive MCMC methods, and thus could be used to draw samples from a given target distribution. As shown by Theorem 1, the distribution of those samples will asymptotically converge to the target distribution.

We test AEMC on a five-dimensional mixture Gaussian distribution

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

where $\mathbf{0} = (0, 0, 0, 0, 0)$ and $\mathbf{5} = (5, 5, 5, 5, 5)$. This example is used in Liang and Wong (2001). Since in this paper we assume the sample space to be bounded, we set the sample space to be $[-10, 10]^5$ here. The distance between the two modes is $5\sqrt{5}$, which makes it difficult to jump from one mode to another. We compare the performance of AEMC with the Metropolis algorithm and EMC. Each algorithm

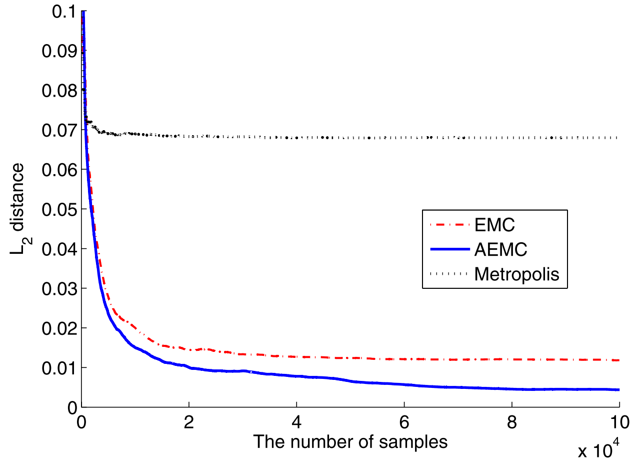

**Fig. 11** Convergence rate of different algorithms

was used to obtain $10^5$ samples and all numerical results were averages of 10 runs.

The Metropolis algorithm was applied with a uniform proposal distribution $U[x-2, x+2]^5$. The acceptance rate was .22. The Metropolis algorithm could not escape from the mode in which it started. We then compare AEMC with EMC. We only look at samples of the first dimension, since each dimension is independent of each other. Since the true histogram of the distribution is known, we can calculate the $L^2$ distance between the estimated mass vector and the true distribution. Specifically, we divide the interval $[-10, 10]$ into 40 intervals (with a resolution of .5), and we can calculate the true and estimated probability mass respectively in each of the intervals.

All EMC related parameters are set following Liang and Wong (2001). In AEMC, we set $h = 25\%$ so that samples from both modes can be obtained. If $h$ is too small, AEMC will focus only on the peaks of the function and thus only samples around the mode $\mathbf{5}$ can be obtained (this is because the probability of sampling around the mode $\mathbf{5}$ is twice as large as the mode $\mathbf{0}$). In EMC, we employ the mutation and crossover operators used in Liang and Wong (2001). The acceptance rates of mutation and crossover operators were .22 and .44, respectively. In AEMC, the acceptance rates of mutation, crossover, and data-mining operators were .23, .54, and .10, respectively. The Fig. 11 shows the $L^2$ distance versus the number of samples for the three methods in comparison. AEMC converges faster than EMC and the Metropolis algorithm, and its sampling quality is far better than the Metropolis algorithm and it also achieves better sampling quality than EMC.

## 5 Conclusions

In this paper, we have presented an AEMC algorithm for optimization problems with "black-box" objective function, which are often encountered in engineering designs (e.g., a sensor placement problem in an assembly process). Our experience indicates that hybridizing a predictive model with an evolutionary Monte Carlo method could improve the convergence rate for an optimization procedure. We have also shown that the algorithm maintains the ergodicity, implying its convergence to the global optimum. Numerical studies are used to compare the proposed AEMC method with other alternatives. All methods in comparison are used to solve a sensor placement problem and to optimize a Griewank function. In these studies, AEMC outperforms other alternatives and shows a much enhanced convergence rate.

This paper focuses mainly on the application of AEMC in solving optimization problems. Yet considering that the AEMC algorithm is an adaptive MCMC method, it should also be useful for sampling from complicated probability distributions. The EMC algorithm has already been shown to be a powerful tool as a sampling method. We believe that the data-mining component could further improve the convergence rate to the target distribution without destroying the ergodicity of the algorithm. We demonstrated the effectiveness of AEMC for sampling purposes using a mixture Gaussian distribution.

In the current version of AEMC, we use CART as the metamodeling method; consequently the sample space is sliced into rectangles. When the function surface is complex, a rectangular partition may not be sufficient. A more sophisticated partition may be required. However, a good replacement for CART may not be straightforward to find because any viable candidate must be computationally efficient so as not to slow down the optimization process. Some other data-mining or predictive modeling methods, such as neural networks, may have more "learning" power, but they are computationally much more expensive and are therefore less likely to be a good candidate for the AEMC algorithm.

**Acknowledgements** The authors would like to gratefully acknowledge financial support from the NSF under grants CMMI-0348150 and CMMI-0726939. The authors also appreciate the editors and the referees for their valuable comments and suggestions.

## Appendix

We first describe the crossover, mutation, and exchange operators used in the EMC algorithm in Appendix A. In Appendix B, we then give a brief summary of the published results on the convergence of adaptive MCMC algorithms. In Appendix C, we prove the ergodicity of the AEMC algorithm.

## Appendix A: Operators in the EMC algorithm

### A.1 Crossover

From the current population $\boldsymbol{x}^{(k)} = \{x_1^{(k)}, \dots, x_n^{(k)}\}$, we first select one parental pair, say $x_i^{(k)}$ and $x_j^{(k)}$ $(i \neq j)$. The first parental sample is chosen according to a roulette wheel procedure with Boltzmann weights. Then the second parental sample is chosen randomly from the rest of the population. So the probability of selecting the pair $(x_i^{(k)}, x_j^{(k)})$ is

$$P((x_i^{(k)}, x_j^{(k)})|\boldsymbol{x}^{(k)}) = \frac{1}{(n-1)G(\boldsymbol{x}^{(k)})} \times \left[\exp\{-H(x_i^{(k)})/\tau_s\} + \exp\{-H(x_j^{(k)})/\tau_s\}\right], \tag{8}$$

where $G(\boldsymbol{x}^{(k)}) = \sum_{i=1}^n \exp\{-H(x_i^{(k)})/\tau_s\}$, and $\tau_s$ is the selection temperature.

Two offsprings are generated by some crossover operator, and the offspring with a smaller fitness value is denoted as $y_j$ and the other is $y_i$. All the crossover operators used in genetic algorithm, e.g., 1-point crossover, 2-point crossover and real crossover, could be used here. Then the new population $\boldsymbol{y} = \{x_1^{(k)}, \dots, y_i, \dots, y_j, \dots, x_n^{(k)}\}$ is accepted with probability min(1,$r_c$),

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

where $T(\cdot|\cdot)$ is the transition probability between populations, and $T(\boldsymbol{y}|\boldsymbol{x}) = P((x_i, x_j)|\boldsymbol{x}) \cdot P((y_i, y_j)|(x_i, x_j))$ for any two populations $\boldsymbol{x}$ and $\boldsymbol{y}$. If the proposal is accepted, the population $\boldsymbol{x}^{(k+1)} = \boldsymbol{y}$, otherwise $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$. Note that all the crossover operators are symmetric, i.e., $P((y_i, y_j)|(x_i, x_j)) = P((x_i, x_j)|(y_i, y_j))$. So $T(\boldsymbol{x}|\boldsymbol{y})/T(\boldsymbol{y}|\boldsymbol{x}) = P((y_i, y_j)|\boldsymbol{y})/P((x_i, x_j)|\boldsymbol{x})$, which can be calculated according to (8).

Following the above selection procedure, samples with better $H(\cdot)$ values have a higher probability to be selected. Offspring generated by these parents will likely to be good as well. In other words, the offspring have learned from the good parents. So crossover operator allows us to construct better proposal distributions, and new samples generated by it are more likely to have better objective $H(\cdot)$ values.

### A.2 Mutation

A sample, say $x_i^{(k)}$, is randomly selected from the current population $\boldsymbol{x}^{(k)}$, then mutated to a new sample $y_i$ by reversing the values of some randomly chosen bits. Then the new population $\boldsymbol{y} = \{x_1^{(k)}, \dots, y_i, \dots, x_n^{(k)}\}$ is accepted with probability min(1, $r_m$),

$$r_m = \frac{f(\boldsymbol{y})}{f(\boldsymbol{x}^{(k)})}\frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x}^{(k)})} = \exp\{-(H(y_i) - H(x_i^{(k)}))/t_i\}\frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{y})}{T(\boldsymbol{y}|\boldsymbol{x}^{(k)})}.$$

The 1-point, 2-point, and uniform mutation are all symmetric operators, and thus $T(\boldsymbol{y}|\boldsymbol{x}^{(k)}) = T(\boldsymbol{x}^{(k)}|\boldsymbol{y})$. If the proposal is accepted, the population $\boldsymbol{x}^{(k+1)} = \boldsymbol{y}$, otherwise $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$.

### A.3 Exchange

Given the current population $\boldsymbol{x}^{(k)}$ and the temperature ladder $\boldsymbol{t}$, we try to change $(\boldsymbol{x}^{(k)}, \boldsymbol{t}) = (x_1^{(k)}, t_1, \dots, x_i^{(k)}, t_i, \dots, x_j^{(k)}, t_j, \dots, x_n^{(k)}, t_n)$ to $(\boldsymbol{x}', \boldsymbol{t}) = (x_1^{(k)}, t_1, \dots, x_j^{(k)}, t_i, \dots, x_i^{(k)}, t_j, \dots, x_n^{(k)}, t_n)$. The new population $\boldsymbol{x}'$ is accepted with probability min(1,$r_e$), where

$$r_e = \frac{f(\boldsymbol{x}')}{f(\boldsymbol{x}^{(k)})}\frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{x}')}{T(\boldsymbol{x}'|\boldsymbol{x}^{(k)})} = \exp\left\{(H(x_i^{(k)}) - H(x_j^{(k)}))\left(\frac{1}{t_i} - \frac{1}{t_j}\right)\right\}\frac{T(\boldsymbol{x}^{(k)}|\boldsymbol{x}')}{T(\boldsymbol{x}'|\boldsymbol{x}^{(k)})}.$$

If this proposal is accepted, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}'$, otherwise $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$. Typically, the exchange is performed only on states with neighboring temperature values, i.e., $|i - j| = 1$. Let $p(x_i^{(k)})$ be the probability that $x_i^{(k)}$ is chosen to exchange with another state, and $w(x_j^{(k)}|x_i^{(k)})$ be the probability that $x_j^{(k)}$ is chosen to exchange with $x_i^{(k)}$. So $j = i \pm 1$, and $w(x_{i+1}^{(k)}|x_i^{(k)}) = w(x_{i-1}^{(k)}|x_i^{(k)}) = .5$ and $w(x_2^{(k)}|x_1^{(k)}) = w(x_{n-1}^{(k)}|x_n^{(k)}) = 1$. The transition probability $T(\boldsymbol{x}'|\boldsymbol{x}^{(k)}) = p(x_i^{(k)}) \cdot w(x_j^{(k)}|x_i^{(k)}) + p(x_j^{(k)}) \cdot w(x_i^{(k)}|x_j^{(k)})$, and thus $T(\boldsymbol{x}'|\boldsymbol{x}^{(k)}) = T(\boldsymbol{x}^{(k)}|\boldsymbol{x}')$.

## Appendix B: Published results on the convergence of adaptive MCMC algorithms

In this section, we briefly review the results presented in Roberts and Rosenthal (2007). Let $f(\cdot)$ be a target probability distribution on a state space $\mathcal{X}^n$ with $\mathcal{B}_{\mathcal{X}^n} = \mathcal{B}_{\mathcal{X}} \times \cdots \times \mathcal{B}_{\mathcal{X}}$ being the $\sigma$-algebra generated by measurable rectangles. Let $\{K_\gamma\}_{\gamma \in \mathcal{Y}}$ be a collection of Markov chain kernels on $\mathcal{X}^n$, each of which has $f(\cdot)$ as a stationary distribution: $(fK_\gamma)(\cdot) = f(\cdot)$. Assume $K_\gamma$ is $\phi$-irreducible and aperiodic, and we have that $K_\gamma$ is ergodic for $f(\cdot)$, i.e., for all $\boldsymbol{x} \in \mathcal{X}^n$, $\lim_{k\to\infty} \|K_\gamma^k(\boldsymbol{x}, \cdot) - f(\cdot)\| = 0$, where $\|\mu(\cdot) - \nu(\cdot)\| = \sup_{\boldsymbol{B}\in\mathcal{B}_{\mathcal{X}^n}} |\mu(\boldsymbol{B}) - \nu(\boldsymbol{B})|$ is the usual total variation distance. Note that $\boldsymbol{B} = B_1 \times \cdots \times B_n$ is a measurable rectangle in $\mathcal{X}^n$, each $B_i \in \mathcal{B}_{\mathcal{X}}$.

For $k = 0, 1, 2, \ldots$, we have a $\mathcal{X}^n$-valued random variable $\boldsymbol{X}^{(k)}$ representing the state of the Markov chain at iteration $k$, and $\mathcal{Y}$-values random variable $\Gamma^{(k)}$ representing the kernel to be used when updating $\boldsymbol{X}^{(k)}$ to $\boldsymbol{X}^{(k+1)}$. Let $A^{(k)}((\boldsymbol{x}, \gamma), \boldsymbol{B}) = \mathbf{P}[\boldsymbol{X}^{(k)} \in \boldsymbol{B} | \boldsymbol{X}^{(0)} = \boldsymbol{x}, \Gamma^{(0)} = \gamma]$, $\boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}$, which represents the conditional probability of $\boldsymbol{X}^{(k)}$ for the adaptive MCMC, given the initial conditions $\boldsymbol{X}^{(0)} = \boldsymbol{x}$ and $\Gamma^{(0)} = \gamma$. Then let

$$T(\boldsymbol{x}, \gamma, k) = \| A^{(k)}((\boldsymbol{x}, \gamma), \cdot) - f(\cdot) \| \equiv \sup_{\boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}} |A^{(k)}((\boldsymbol{x}, \gamma), \boldsymbol{B}) - f(\boldsymbol{B})|$$

denote the total variation distance between the $f(\cdot)$ and the distribution of the adaptive MCMC algorithm at iteration $k$. Then the adaptive algorithm is called ergodic if

$$\lim_{k \to \infty} T(\boldsymbol{x}, \gamma, k) = 0, \quad \text{for all } \boldsymbol{x} \in \mathcal{X}^n \text{ and } \gamma \in \mathcal{Y}.$$

The following states three conditions that are used to prove the ergodicity of an adaptive MCMC algorithm.

(A$_1$) (*Strongly aperiodic minorisation condition*) There is a $\boldsymbol{C} \in \mathcal{X}^n$, $\varphi > 0$, and for each $\gamma \in \mathcal{Y}$, there exists a probability measure $\nu_\gamma(\cdot)$ on $\mathcal{X}^n$ such that

$$K_\gamma(\boldsymbol{x}, \boldsymbol{B}) \ge \varphi \nu_\gamma(\boldsymbol{B}), \quad \forall \boldsymbol{x} \in \boldsymbol{C}, \ \forall \boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}. \tag{9}$$

(A$_2$) (*Geometric drift condition*) There is $V : \mathcal{X}^n \to [1, \infty)$, $0 < \lambda < 1$, $b < \infty$, and $\sup_{\boldsymbol{C}} V = v < \infty$ such that for each $\gamma \in \mathcal{Y}$

$$K_\gamma V(\boldsymbol{x}) \le \lambda V + bI(\boldsymbol{x} \in \boldsymbol{C}), \quad \forall \boldsymbol{x} \in \mathcal{X}^n \tag{10}$$

where $K_\gamma V(\boldsymbol{x}) = \int_{\mathcal{X}^n} K_\gamma(\boldsymbol{x}, \boldsymbol{y}) V(\boldsymbol{y}) d\boldsymbol{y}$.

(A$_3$) (*Diminishing adaptation condition*) The diminishing adaptation condition holds if

$$\lim_{k \to \infty} \sup_{\boldsymbol{x} \in \mathcal{X}^n} \| K_{\Gamma_{k+1}}(\boldsymbol{x}, \cdot) - K_{\Gamma_k}(\boldsymbol{x}, \cdot) \| = 0. \tag{11}$$

We then have:

**Theorem 2** *Consider an adaptive MCMC algorithm with a family of Markov chain kernels $\{K_\gamma\}_{\gamma \in \mathcal{Y}}$ satisfying the conditions* (A$_1$), (A$_2$) *and* (A$_3$), *and $\boldsymbol{E}[V(\boldsymbol{x})] < \infty$. Then the adaptive algorithm is ergodic.*

## Appendix C: Proof of ergodicity of AEMC

As stated in Theorem 1, we have the following:

*If $\mathcal{X}$ is compact, $0 < r < 1$, and $\lim_{k \to \infty} P_k = 0$, then AEMC is ergodic, i.e., the samples $\boldsymbol{x}^{(k)}$ converge in distribution to $f(\boldsymbol{x})$.*

*Proof* Denote the Markov chain kernel in the data-mining mode at iteration $k$ by $K_{\Gamma_k}^{(1)}(\boldsymbol{x}, \cdot)$. To prove ergodicity, we need to prove the conditions (A$_1$), (A$_2$) and (A$_3$) for the $K_{\Gamma_k}^{(1)}(\boldsymbol{x}, \cdot)$, $k = 0, 1, 2, \ldots$. For notational simplicity, we drop the subscript $k$ in the proof, denoting $K_{\Gamma_k}^{(1)}$ by $K_\Gamma^{(1)}$ and $\boldsymbol{x}^{(k)}$ by $\boldsymbol{x}$. We denote the proposal density learned by CART by $q_\Gamma(\cdot)$ afterwards. Since we use the Metropolis-within-Gibbs procedure in the data mining mode, we have

$$\begin{aligned} K_\Gamma^{(1)}(\boldsymbol{x}, \boldsymbol{y}) = {} & K_\Gamma^{(1,1)}((x_1, x_2, \ldots, x_n), (y_1, x_2, \ldots, x_n)) \times \cdots \\ & \times K_\Gamma^{(1,n)}((y_1, \ldots, y_{n-1}, x_n), \\ & (y_1, \ldots, y_{n-1}, y_n)), \end{aligned} \tag{12}$$

where $K_\Gamma^{(1,i)}(\cdot, \cdot)$ is the Metropolis-Hastings kernel for the transition of the $i$-th sample of the population and can be written as

$$\begin{aligned} & K_\Gamma^{(1,i)}(x_i, y_i | \boldsymbol{\xi}_i) \\ & \triangleq K_\Gamma^{(1,i)}((y_1, \ldots, y_{i-1}, x_i, \ldots, x_n), \\ & \qquad (y_1, \ldots, y_i, x_{i+1}, \ldots, x_n)) \\ & = s_\Gamma(x_i, y_i | \boldsymbol{\xi}_i) + I(x_i = y_i) \left( 1 - \int_{\mathcal{X}} s_\Gamma(x_i, z | \boldsymbol{\xi}_i) dz \right). \end{aligned} \tag{13}$$

Here $\boldsymbol{\xi}_i = (y_1, \ldots, y_{i-1}, x_{i+1}, \ldots, x_n)$ denotes the collection of the fixed samples in the transition, $s_\Gamma(x_i, y_i | \boldsymbol{\xi}_i) = q_\Gamma(y_i) \min\{1, \frac{p(y_i | \boldsymbol{\xi}_i) q_\Gamma(x_i)}{p(x_i | \boldsymbol{\xi}_i) q_\Gamma(y_i)}\}$, and $p(z | \boldsymbol{\xi})$ is the conditional density of $z$ given the other components $\boldsymbol{\xi}$.

Since we have assumed that $\mathcal{X}^n$ is compact, it is natural to assume that $f(\boldsymbol{x})$ is bounded away from 0 and $\infty$ on the space $\mathcal{X}^n$. As long as $0 < r < 1$, we have that the proposal $q_\Gamma(z)$ is bounded away from 0 due to (6), and then we have the minorisation condition, i.e.,

$$\omega^* = \sup_{y \in \mathcal{X}} \frac{p(y | \boldsymbol{\xi})}{q_\Gamma(y)} < \infty. \tag{14}$$

And then

$$\begin{aligned} & K_\Gamma^{(1,i)}(z, B_i | \boldsymbol{\xi}_i) \\ & = \int_{B_i} s_\Gamma(z, y | \boldsymbol{\xi}_i) dy \\ & \quad + I(z \in B_i) \left( 1 - \int_{\mathcal{X}} s_\Gamma(z, w | \boldsymbol{\xi}_i) dw \right) \\ & \ge \int_{B_i} q_\Gamma(y) \min\left\{ 1, \frac{p(y | \boldsymbol{\xi}_i) q_\Gamma(z)}{p(z | \boldsymbol{\xi}_i) q_\Gamma(y)} \right\} dy \\ & = \int_{B_i} \min\left\{ q_\Gamma(y), \frac{p(y | \boldsymbol{\xi}_i) q_\Gamma(z)}{p(z | \boldsymbol{\xi}_i)} \right\} dy \end{aligned}$$

$$\geq \int_{B_i} \min\left\{q_\Gamma(y), \frac{p(y|\boldsymbol{\xi}_i)}{\omega^*}\right\} dy \quad \text{(by (14))}$$

$$= \int_{B_i} \frac{p(y|\boldsymbol{\xi}_i)}{\omega^*} dy \quad \text{(by definition of } \omega^*)$$

$$\geq \int_{B_i} \frac{p_i^*(y)}{\omega^*} dy$$

$$\text{(by defining } p_i^*(y) = \inf_{\boldsymbol{\xi}_i' \in \mathcal{X}^{n-1}} p(y|\boldsymbol{\xi}_i'))$$

$$= \frac{p_i^*(B_i)}{\omega^*}.$$

Since we have assumed that $f(\boldsymbol{x})$ is bounded away from 0 and $\infty$, $p(y|\xi_i)$, as the conditional density of a component of $\boldsymbol{x}$, is bounded away from 0 and $\infty$, and so are $p_i^*(y)$ and $p_i^*(B_i)$. Therefore, we have the following results:

$$K_\Gamma^{(1)}(\boldsymbol{x}, \boldsymbol{B})$$
$$= \int_{B_1} \cdots \int_{B_n} K_\Gamma^{(1,1)}(x_1, y_1|\boldsymbol{\xi}_1) \times \cdots$$
$$\times K_\Gamma^{(1,n)}(x_n, y_n|\boldsymbol{\xi}_n) dy_1 \cdots dy_n$$
$$\geq \frac{p_n^*(B_n)}{\omega^*} \int_{B_1} \cdots \int_{B_{n-1}} K_\Gamma^{(1,1)}(x_1, y_1|\boldsymbol{\xi}_1) \times \cdots$$
$$\times K_\Gamma^{(1,n-1)}(x_{n-1}, y_{n-1}|\boldsymbol{\xi}_{n-1}) dy_1 \cdots dy_{n-1}$$
$$\cdots$$
$$\geq \prod_{i=1}^n p_i^*(B_i)/(\omega^*)^n.$$

Define $\nu_\Gamma(\boldsymbol{B}) = \prod_{i=1}^n p_i^*(B_i)$ and $\varphi = 1/(\omega^*)^n$, and then we have

$$K_\Gamma^{(1)}(\boldsymbol{x}, \boldsymbol{B}) \geq \varphi \nu_\Gamma(\boldsymbol{B}), \quad \forall \boldsymbol{x} \in \mathcal{X}^n, \ \forall \boldsymbol{B} \in \mathcal{B}_{\mathcal{X}^n}. \tag{15}$$

The equation (15) implies that the condition (A$_1$) is satisfied, and it also implies that $\boldsymbol{C} = \mathcal{X}^n$ is a small set (Mengersen and Tweedie 1996) and that the following condition holds:

$$K_\Gamma^{(1)} V(\boldsymbol{x}) \leq \lambda V(\boldsymbol{x}) + bI(\boldsymbol{x} \in \boldsymbol{C}), \quad \forall \boldsymbol{x} \in \mathcal{X}^n, \tag{16}$$

by choosing $V(\boldsymbol{x}) = 1, 0 < \lambda < 1, b = 1 - \lambda$. Then the equation (16) implies that the condition (A$_2$) is satisfied. Also we have $\boldsymbol{E}[V(\boldsymbol{x})] < \infty$.

Now we prove the Diminishing Adaptation condition for the kernel $K_\Gamma^{(1)}$. Suppose at iteration $k+1$ the newly learned data-mining proposal is $K_{\Gamma^*}^{(1)}$, then

$$K_{\Gamma_{k+1}}^{(1)} = P_{k+1} K_{\Gamma^*}^{(1)} + (1 - P_{k+1}) K_{\Gamma_k}^{(1)}.$$

Then we have $\|K_{\Gamma_{k+1}}^{(1)} - K_{\Gamma_k}^{(1)}\| = P_{k+1} \|K_{\Gamma^*}^{(1)} - K_{\Gamma_k}^{(1)}\|$. Since $\lim_{k\to\infty} P_k = 0$, to prove the equation (11), it suffices to prove that $\|K_{\Gamma^*}^{(1)} - K_{\Gamma_k}^{(1)}\|$ is bounded. Since both $K_{\Gamma^*}^{(1)}$ and $K_{\Gamma_k}^{(1)}$ are Markov chain kernels, we have $\|K_{\Gamma^*}^{(1)} - K_{\Gamma_k}^{(1)}\| \leq 2$, and

$$\|K_{\Gamma_{k+1}}^{(1)} - K_{\Gamma_k}^{(1)}\| = P_{k+1} \|K_{\Gamma^*}^{(1)} - K_{\Gamma_k}^{(1)}\| \leq 2P_{k+1} \to 0.$$

This proves the diminishing adaptation condition.

The proof is complete. □

## References

Andrieu, C., Robert, C.P.: Controlled MCMC for optimal sampling. MCMC Preprint Service. http://www.ceremade.dauphine.fr/~xian/control.ps.gz (2002)
Atchadé, Y.F., Rosenthal, J.S.: On adaptive Markov chain Monte Carlo algorithms. Bernoulli **11**, 815–828 (2005)
Bertsimas, D., Tsitsiklis, J.: Simulated annealing. Stat. Sci. **8**, 10–15 (1993)
Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Chapman & Hall, New York (1984)
Brockwell, A.E., Kadane, J.B.: Identification of regeneration times in MCMC simulation, with application to adaptive schemes. J. Comput. Graph. Stat. **14**, 436–458 (2005)
Brooks, R.R., Ramanathan, P., Sayeed, A.M.: Distributed target classification and tracking in sensor networks. Proc. IEEE **91**, 1163–1171 (2003)
Box, G.E.P., Wilson, K.B.: On the experimental attainment of optimum conditions. J. R. Stat. Soc. B Stat. Methodol. **13**, 1–45 (1951)
Bukkapatnam, S.T.S., Nichols, J.M., Seaver, M., Trickey, S.T., Hunter, M.: A wavelet-based, distortion energy approach to structural health monitoring. Struct. Health Monitor. J. **4**, 247–258 (2005)
Chen, V.C.P., Tsui, K., Barton, R.R., Meckesheimer, M.: A review on design, modeling and applications of computer experiments. IIE Trans. **38**, 273–291 (2006)
Čivilis, A., Jensen, C.S., Pakalnis, S.: Techniques for efficient tracking of road-network-based moving objects. IEEE Trans. Knowl. Data Eng. **17**, 698–712 (2005)
Fang, K., Li, R., Sudjianto, A.: Design and modeling for computer experiments. Chapman & Hall/CRC, Boca Raton (2006)
Geman, S., Geman, D.: Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. IEEE Trans. Pattern Anal. **6**, 721–741 (1984)
Gilks, W.R., Richardson, S., Spiegelhalter, D.J.: Introducing Markov chain Monte Carlo. In: Markov Chain Monte Carlo in Practice, pp. 1–19. Chapman & Hall, London (1995)
Gilks, W.R., Roberts, G.O., Sahu, S.K.: Adaptive Markov chain Monte Carlo through regeneration. J. Am. Stat. Assoc. **93**, 1045–1054 (1998)
Griewank, A.O.: Generalized descent for global optimization. J. Optim. Theory Appl. **34**, 11–39 (1981)
Guikema, S.D., Davidson, R.A., Çağnan, Z.: Efficient simulation-based discrete optimization. In: Ingalls, R.G., Rossetti, M.D., Smith, J.S., Peters, B.A. (eds.) Proceedings of the 2004 Winter Simulation Conference (2004)
Haario, H., Saksman, E., Tamminen, J.: An adaptive metropolis algorithm. Bernoulli **7**, 223–242 (2001)
Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, New York (2001)
Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge (1992)
Huyet, A.L.: Optimization and analysis aid via data-mining for simulated production systems. Eur. J. Oper. Res. **173**, 827–838 (2006)
Jin, J., Shi, J.: State space modeling of sheet metal assembly for dimensional control. J. Manuf. Sci. Eng. **121**, 756–762 (1999)

Liang, F.: A generalized Wang-Landau algorithm for Monte Carlo computation. J. Am. Stat. Assoc. **100**, 1311–1327 (2005)

Liang, F., Liu, C., Carroll, R.J.: Stochastic approximation in Monte Carlo Computation. J. Am. Stat. Assoc. **102**, 305–320 (2007)

Liang, F., Wong, W.H.: Evolutionary Monte Carlo: applications to $C_p$ model sampling and change point problem. Stat. Sin. **10**, 317–342 (2000)

Liang, F., Wong, W.H.: Real-parameter evolutionary Monte Carlo with applications to Bayesian mixture models. J. Am. Stat. Assoc. **96**, 653–666 (2001)

Liu, H., Igusa, T.: Feature-based classification for design optimization. Res. Eng. Des. **17**, 189–206 (2007)

Kim, P., Ding, Y.: Optimal engineering system design guided by data-mining methods. Technometrics **47**, 336–348 (2005)

Liu, C., Ding, Y., Chen, Y.: Optimal coordinate sensor placements for estimating mean and variance components of variation sources. IIE Trans. **37**, 877–889 (2005)

Mandroli, S.S., Shrivastava, A.K., Ding, Y.: A survey of inspection strategy and sensor distribution studies in discrete-part manufacturing processes. IIE Trans. **38**, 309–328 (2006)

Mengersen, K.L., Tweedie, R.L.: Rates of convergence of the Hastings and Metropolis algorithms. Ann. Stat. **24**, 101–121 (1996)

Michalski, R.S.: Learnable evolution model: evolutionary processes guided by machine learning. Mach. Learn. **38**, 9–40 (2000)

Müller, P.: A generic approach to posterior integration and Gibbs sampling. Technical Report, Purdue University, West Lafayette, Indiana (1991)

Neal, R.M.: Bayesian learning for neural networks. Lecture Notes in Statistics, vol. 118. Springer, New York (1996)

Roberts, G.O., Rosenthal, J.S.: General state-space Markov chains and MCMC algorithms. Probab. Surv. **1**, 20–71 (2004)

Roberts, G.O., Rosenthal, J.S.: Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. J. Appl. Probab. **44**, 458–475 (2007)

Roberts, G.O., Tweedie, R.L.: Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. Biometrika **83**, 95–110 (1996)

Rosenthal, J.S.: Minorization conditions and convergence rate for Markov chain Monte Carlo. J. Am. Stat. Assoc. **90**, 558–566 (1995)

Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. **4**, 409–435 (1989)

Schwabacher, M., Ellman, T., Hirsh, H.: Learning to set up numerical optimizations of engineering designs. In: Braha, D. (ed.) Data Mining for Design and Manufacturing, pp. 87–125. Kluwer Academic, Dordrecht (2001)

Simpson, T.W., Peplinski, J., Koch, P.N., Allen, J.K.: On the use of statistics in design and the implications for deterministic computer experiments. In: Design Theory and Methodology—DTM’97, Sacramento, CA, September 14–17, ASME, Paper No. DETC97/DTM-3881 (1997)

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