A Generalized Wang–Landau Algorithm for Monte Carlo Computation

Faming LIANG

Inference for a complex system with a rough energy landscape is a central topic in Monte Carlo computation. Motivated by the successes of the Wang–Landau algorithm in discrete systems, we generalize the algorithm to continuous systems. The generalized algorithm has some features that conventional Monte Carlo algorithms do not have. First, it provides a new method for Monte Carlo integration based on stochastic approximation; second, it is an excellent tool for Monte Carlo optimization. In an appropriate setting, the algorithm can lead to a random walk in the energy space, and thus it can sample relevant parts of the sample space, even in the presence of many local energy minima. The generalized algorithm can be conveniently used in many problems of Monte Carlo integration and optimization, for example, normalizing constant estimation, model selection, highest posterior density interval construction, and function optimization. Our numerical results show that the algorithm outperforms simulated annealing and parallel tempering in optimization for the system with a rough energy landscape. Some theoretical results on the convergence of the algorithm are provided.

KEY WORDS: $1/k$-ensemble sampling; Importance sampling; Markov chain Monte Carlo; Monte Carlo integration; Monte Carlo optimization; Multicanonical algorithm; Wang–Landau algorithm.

1. INTRODUCTION

Let $f(\mathbf{x})$ be a given probability mass/density function expressed in the form

$$f(\mathbf{x}) = \frac{1}{Z_\tau} \exp\{-H(\mathbf{x})/\tau\}, \qquad \mathbf{x} \in \mathcal{X}, \tag{1}$$

where τ is the temperature, Z_τ is the normalizing constant, and $H(\mathbf{x})$ is the energy function, which corresponds to the negative of the log-posterior of $\mathbf{x}$ in Bayesian computation. In statistics we are often interested in two problems: estimating the expectation $E_f h(\mathbf{x})$ for a given function $h(\mathbf{x})$ with $E_f|h(\mathbf{x})| < \infty$, and minimizing the energy function $H(\mathbf{x})$. In the context of Monte Carlo computation, these problems are termed Monte Carlo integration and Monte Carlo optimization.

Monte Carlo algorithms can be divided into two categories according to the distribution from which the samples are drawn, namely simulation algorithms and importance sampling algorithms. Simulation algorithms include the Metropolis–Hastings (MH) algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller 1953; Hastings 1970), the Gibbs sampler (Geman and Geman 1984), the Swendsen–Wang algorithm (Swendsen and Wang 1987), simulated tempering (Marinari and Parisi 1992; Geyer and Thompson 1995), parallel tempering (Geyer 1991), evolutionary Monte Carlo (Liang and Wong 2001), and others. They work by simulating a Markov chain that is able to generate samples from the target distribution $f(\mathbf{x})$. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ be the samples generated by the Markov chain, where the successive samples may be highly correlated. The $E_f h(\mathbf{x})$ can be estimated by

$$\widetilde{E_f h(\mathbf{x})} = \frac{1}{n} \sum_{i=1}^{n} h(\mathbf{x}_i). \tag{2}$$

The ergodicity of the Markov chain ensures that as $n \to \infty$, $\widetilde{E_f h(\mathbf{x})} \to E_f h(\mathbf{x})$ almost surely (Tierney 1994; Roberts 1996). In simulation from $f(\mathbf{x})$, the $H(\mathbf{x})$'s can often be minimized simultaneously; for example, simulated tempering has been successfully used in optimizing protein structures (Hansmann and Okamoto 1997).

In importance sampling, one does not draw samples from $f(\mathbf{x})$ directly but rather from a trial distribution $\pi(\mathbf{x})$ that has a much "smoother" energy landscape than $f(\mathbf{x})$ but also has key characteristics of $f(\mathbf{x})$. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n$ denote samples drawn from $\pi(\mathbf{x})$. The $E_f h(\mathbf{x})$ can be estimated by

$$\widehat{E_f h(\mathbf{x})} = \frac{\sum_{i=1}^{n} h(\mathbf{x}_i) w_i}{\sum_{i=1}^{n} w_i}, \tag{3}$$

where $w_i = f(\mathbf{x}_i)/\pi(\mathbf{x}_i)$ is the importance weight of $\mathbf{x}_i$. In simulation from $\pi(\mathbf{x})$, $H(\mathbf{x})$ can also be minimized. For example, simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983) minimizes $H(\mathbf{x})$ by simulating from a sequence of trial distributions with the temperature scaled from high to low. Other algorithms of this category include dynamically weighted importance sampling (DWIS) (Liang 2002), the multicanonical algorithm (Berg and Neuhaus 1991), $1/k$-ensemble sampling (Hesselbo and Stinchcombe 1995), and the Wang–Landau (WL) algorithm (Wang and Landau 2001). In DWIS $\pi(\mathbf{x})$ is defined implicitly, and the weight is itself a random variable. The complexity of the weight control scheme makes the algorithm not very user-friendly. The multicanonical algorithm and the WL algorithm are essentially the same (as reviewed in Sec. 2). They seek to sample from a trial distribution under which the energy variable $U = H(\mathbf{x})$ is approximately uniformly distributed. They are very useful in optimization problems, especially for a problem that has a rough energy landscape. The energy landscape of a problem or distribution refers to the image of the sample space $\mathcal{X}$ under the corresponding energy function $H(\mathbf{x})$. These two algorithms have been applied successfully to some hard optimization problems, including traveling salesman problems (Lee and Cho 1994) and protein folding (Hansmann and Okamoto 1997; Rathore and de Pablo 2002). Despite the successes in Monte Carlo optimization, the use of these two algorithms in Monte Carlo integration is limited only to discrete distributions, say, the Potts model (Wang and Landau 2001;

Faming Liang is Assistant Professor, Department of Statistics, Texas A&M University, College Station, TX 77843 (E-mail: *fliang@stat.tamu.edu*). Liang's research was supported in part by National Science Foundation grant DMS-04-05748. The author thanks F. J. Samaniego, an associate editor, and the referees for their constructive comments and suggestions that led to significant improvement of this article.

© 2005 American Statistical Association
Journal of the American Statistical Association
December 2005, Vol. 100, No. 472, Theory and Methods
DOI 10.1198/016214505000000259

Yamaguchi and Okabe 2001), for estimating the spectral density (defined in Sec. 2.1) of the distribution. Applying them to continuous distributions will lead to some uninterpretable or uninteresting results. This is similar to the $1/k$-ensemble sampling algorithm.

In this article we generalize the WL algorithm to continuous systems. The generalized algorithm provides a new method, in addition to (2) and (3), for Monte Carlo integration. The generalized algorithm has also introduced a user-specified parameter such that it can be tuned to sample more frequently from some specific area of the sample space. The generalized algorithm can be used in many problems of Monte Carlo integration and optimization, including normalizing constant estimation, highest posterior density (HPD) interval construction, model selection, and function optimization. Numerical results show that it outperforms other advanced Monte Carlo algorithms, such as simulated annealing and parallel tempering, for the problem with a rough energy landscape.

The article is organized as follows. Section 2 briefly reviews the multicanonical and related algorithms. Section 3 describes the generalized Wang–Landau (GWL) algorithm and gives some theoretical results on the algorithm. Section 4 compares the GWL algorithm with the multicanonical and WL algorithms through a numerical example. Section 5 demonstrates the use of the GWL algorithm in Monte Carlo integration. Section 6 demonstrates the use of the GWL algorithm in Monte Carlo optimization. Section 7 concludes the article with a brief discussion.

2. MULTICANONICAL AND RELATED ALGORITHMS: A LITERATURE REVIEW

Suppose that our target distribution is $f(\mathbf{x})$ as defined in (1). As in earlier works (Berg and Neuhaus 1991; Hesselbo and Stinchcombe 1995; Wang and Landau 2001), we assume that the sample space $\mathcal{X}$ only contains a finite number of sample points and the energy function $H(\mathbf{x})$ only takes a finite number of distinct values. Let $\{u_1, \dots, u_m\}$ be a set of real numbers containing all possible values of $H(\mathbf{x})$. We describe these algorithms in the following sections.

2.1 The Multicanonical Algorithm

As mentioned in Section 1, the multicanonical algorithm seeks to sample from a trial distribution $\pi(\mathbf{x})$ under which the energy variable $U = H(\mathbf{x})$ is approximately uniformly distributed. With a slight abuse of notations, we denote the marginal distribution of U by $f_U(u)$,

$$f_U(u) = \frac{1}{Z_\tau}\Omega(u)e^{-u/\tau}, \tag{4}$$

where $\Omega(u) = \#\{\mathbf{x}: H(\mathbf{x}) = u\}$ is called the spectral density (or the density of states) of $f(\mathbf{x})$. If the samples are drawn from

$$\pi(\mathbf{x}) \propto e^{-V(H(\mathbf{x}))},$$

where $V(H(\mathbf{x})) = \log \Omega(H(\mathbf{x}))$, then the marginal distribution of U under $\pi(\mathbf{x})$ is $\pi_U(u) \propto 1$; that is, U is uniformly distributed. Therefore, the key step of the multicanonical algorithm is to estimate $\Omega(u)$.

The initial estimate of $\Omega(u)$ can be obtained as follows through a short simulation from $f(\mathbf{x})$. Let $X_1, \dots, X_N$ denote the Markov chain Monte Carlo (MCMC) samples drawn from $f(\mathbf{x})$, and let $N(u_i) = \#\{\mathbf{x}_j: H(\mathbf{x}_j) = u_i\}$ denote the number of samples with energy u_i. As $N \to \infty$, we have

$$\frac{N(u_i)}{N} \to \frac{1}{Z_\tau}\Omega(u_i)e^{-u_i/\tau}.$$

Thus $\Omega(u_i)$ can be estimated by

$$\hat{\Omega}(u_i) = \frac{(N(u_i) + c_i)e^{u_i/\tau}}{\sum_{j=1}^m (N(u_j) + c_j)e^{u_j/\tau}}, \tag{5}$$

where the c_j's serve as "prior counts" to smooth out the estimate in Ω. Let $\hat{V}_s(H(\mathbf{x}))$ denote the estimate of $V(H(\mathbf{x}))$ obtained at stage s. Set $\hat{V}_0(u) = \log \hat{\Omega}(u)$; then $\hat{V}_s(u)$ can be updated iteratively as follows:

a. Sample $\mathbf{x}$ sufficiently long according to the current trial distribution $\pi_s(\mathbf{x}) \propto e^{-\hat{V}_s(H(\mathbf{x}))}$. The sampling can be done by a conventional simulation algorithm, say, the MH algorithm.

b. Update $\hat{V}_{s+1}(u_i) = -\log(a_s) + \hat{V}_s(u_i) + \log(N(u_i) + c_i)$ for $i = 1, \dots, m$, where a_s is introduced to ensure that $\hat{V}_{s+1}(u)$ is scaled properly. For example, a_s can be determined by imposing on the $\hat{V}_{s+1}(u_j)$'s the constraint $\sum_{j=1}^m \exp(\hat{V}_{s+1}(u_j)) = \sum_{j=1}^m \Omega(u_j)$, where $\sum_{j=1}^m \Omega(u_j)$ is assumed to be known a priori.

The recursive formula in step b enforces a "free" random walk in the energy space by penalizing moving to and staying at the energy that was overvisited in the previous stage.

2.2 $1/k$-Ensemble Sampling

Similar to the multicanonical algorithm, $1/k$-ensemble sampling (Hesselbo and Stinchcombe 1995) suggests the following trial distribution for simulation:

$$\pi(\mathbf{x}) \propto \frac{1}{k(H(\mathbf{x}))},$$

where $k(H(\mathbf{x})) = \sum_{u' \le H(\mathbf{x})} \Omega(u')$ is the number of states with energies up to and including $H(\mathbf{x})$. Because $k(\cdot)$ is an increasing function of energy, it will lead to a random walk in the space of energy, but with more weights toward low-energy regions. Improvement over the multicanonical algorithm is observed in the ergodicity of simulations for the Ising model and traveling salesman problems.

In practice, the function $k(u)$ is unknown a priori. It can be estimated with the same iterative strategy as described in the previous section. Obviously, with a good estimation to $\Omega(u)$, one would be able to get a good estimation to $k(u)$, and vice versa.

2.3 The Wang–Landau Algorithm

The WL algorithm provides a much convenient method for estimating the spectral density $\Omega(u)$. Hence it can be considered an innovative implementation for the multicanonical algorithm. But it goes beyond that. In the multicanonical algorithm, the random walk tends to be blocked by the edge of the already-visited area; in addition, it takes a long time to traverse an area because of the general features of a random walk. The WL algorithm succeeds in removing these problems by timely penalizing moving to and staying at the energy that has been visited many times.

The simulation of the WL algorithm consists of several stages. The first stage starts with the initial estimates $\hat{\Omega}(u_1) = \cdots = \hat{\Omega}(u_m) = 1$ and a sample $\mathbf{x}_0$ drawn from $\mathcal{X}$ at random, and iterates between the following steps:

a. Propose a new configuration $\mathbf{x}^*$ according to a symmetric proposal distribution T, that is, $T(\mathbf{x}^*|\mathbf{x}_k) = T(\mathbf{x}_k|\mathbf{x}^*)$.

b. Accept $\mathbf{x}^*$ with probability $\min\{\frac{\hat{\Omega}(H(\mathbf{x}_k))}{\hat{\Omega}(H(\mathbf{x}^*))}, 1\}$. If it is accepted, then set $\mathbf{x}_{k+1} = \mathbf{x}^*$ and $\hat{\Omega}(H(\mathbf{x}_{k+1})) \leftarrow \hat{\Omega}(H(\mathbf{x}^*)) \times (1+\delta)$; otherwise, set $\mathbf{x}_{k+1} = \mathbf{x}_k$ and $\hat{\Omega}(H(\mathbf{x}_k)) \leftarrow \hat{\Omega}(H(\mathbf{x}_k)) \times (1+\delta)$, where $\delta > 0$ is a modification factor.

The algorithm iterates until a flat histogram has been produced in the energy space. Wang and Landau (2001) consider a histogram to be flat if the sampling frequency for each of the u_i's is not less than 80% of the average sampling frequency. Once this condition is satisfied, the estimated $\hat{\Omega}(u_i)$'s and the current sample $\mathbf{x}_k$ are passed on to the next stage as initial values, the modification factor is reduced to a smaller value, and the sampler collector is resumed. The next-stage simulation is then started, continuing until the new histogram is flat again. Wang and Landau (2001) recommended that δ should decrease in the scheme $\delta_{\text{new}} = \sqrt{1+\delta_{\text{old}}} - 1$, which is approximately equivalent to the geometric scheme $\delta_{\text{new}} = .5\delta_{\text{old}}$ when δ is small. The process is repeated until δ is smaller than some specified value, say, 10^{-8}. As $\delta \to 0$, the foregoing process satisfies the detailed balance condition approximately. Based on this, Wang and Landau (2001) claimed that $\hat{\Omega}(u)$ will converge to $\Omega(u)$ as $\delta \to 0$ and $k \to \infty$.

We note that the multicanonical algorithm and $1/k$-ensemble sampling have been applied by Hannsmann and Okamoto (1997) to optimization for a continuous system. They discretize the energy as $-\infty < u_1 < \cdots < u_m < \infty$, and redefine $N(u_i)$ as the number of samples with energy between u_{i-1} and u_i; that is, $N(u_i) = \#\{\mathbf{x}_j : u_{i-1} < H(\mathbf{x}_j) \le u_i\}$. In this case $\Omega(u_i)$ is no longer the spectral density of the system, but rather a number proportional to $\int_{E_i} d\mathbf{x}$, the hypervolume of the subregion $E_i = \{\mathbf{x} : u_{i-1} < H(\mathbf{x}) \le u_i\}$. The quantity $\int_{E_i} d\mathbf{x}$ is seldom of interest to us.

3. GENERALIZED WANG–LANDAU ALGORITHM

The GWL algorithm generalizes over the WL algorithm in two respects. First, it extends the use of the WL algorithm to Monte Carlo integration for continuous systems. The extension is done by specifying a working function $\psi(\mathbf{x})$, $\mathbf{x} \in \mathcal{X}$; partitioning the sample space into a finite number of disjoint subregions, $E_1, \ldots, E_m$; and attaching a different weight, $g(E_i)$ (defined later), to each of the subregions. For example, if we are interested in estimating the expectation $E_f h(\mathbf{x})$, where $h(\mathbf{x}) > 0$ and $E_f|h(\mathbf{x})| < \infty$, then we can set $\psi(\mathbf{x}) = h(\mathbf{x}) f(\mathbf{x})$. We discuss the case for a general function $h(\mathbf{x})$ in Section 5. The sample space can be partitioned using a method similar to that used by Hannsmann and Okamoto (1997). For example, the sample space can be partitioned into $E_1 = \{\mathbf{x} : H(\mathbf{x}) \le u_1\}$, $E_2 = \{\mathbf{x} : u_1 < H(\mathbf{x}) \le u_2\}, \ldots$, and $E_m = \{\mathbf{x} : H(\mathbf{x}) > u_{m-1}\}$ for a set of given energy values $u_1, \ldots, u_{m-1}$. The partition can be made according to any function of $\mathbf{x}$, say, a component of $\mathbf{x}$, instead of the energy function only. Second, a user-specified parameter ρ ($\rho \ge 0$) is introduced into the GWL algorithm to control the sampling frequency of each of the subregions. The GWL algorithm can be tuned to sample more frequently from some subregions of interest by tuning the value of ρ, whereas the WL algorithm can sample only from each of the subregions equally. With the parameter ρ, the $g(E_i)$'s can be defined as

$$g(E_1) = \int_{E_1} \psi(\mathbf{x})\, d\mathbf{x} \qquad \text{and}$$
$$g(E_i) = \int_{E_i} \psi(\mathbf{x})\, d\mathbf{x} + \rho g(E_{i-1}) \quad \text{for } i = 2, \ldots, m, \tag{6}$$

which work as analogies of the $\Omega(u_i)$'s of the WL algorithm. The $g(E_i)$'s have very meaningful statistical interpretations. $g(E_i) - \rho g(E_{i-1})$ [setting $g(E_0) = 0$] is the normalizing constant of the truncated distribution of $\psi(\mathbf{x})$ on the subregion E_i; $g(E_1), g(E_2) - \rho g(E_1), \ldots, g(E_m) - \rho g(E_{m-1})$ forms a histogram estimate for the marginal density of $f(\mathbf{x})$ in the space of subregions; and $\sum_{i=1}^{n} [g(E_i) - \rho g(E_{i-1})] = \int_{\mathcal{X}} \psi(\mathbf{x})\, d\mathbf{x}$. Therefore, the key step of the GWL algorithm is to estimate the $g(E_i)$'s.

Let $T(\mathbf{y}|\mathbf{x})$ be a global proposal distribution, where $\mathbf{x}$ denotes the current state and $\mathbf{y}$ denotes the proposed one. $T(\mathbf{y}|\mathbf{x})$ is called a global proposal distribution if $T(\mathbf{y}|\mathbf{x}) > 0$ is true for any pair $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}_0 \times \mathcal{X}_0$, where $\mathcal{X}_0 = \{\mathbf{x} : \psi(\mathbf{x}) > 0\}$ is a reduced sample space induced by $\psi(\mathbf{x})$. The global proposal distribution is used in the GWL algorithm to guarantee that $\mathcal{X}_0$ can be fully explored in simulation regardless of its shape. Many of the commonly used proposal distributions are global, including the normal, Cauchy and Student t-distributions. Note that in the GWL algorithm, the proposal distribution is not necessarily symmetric. This makes the GWL algorithm much more flexible than the WL algorithm in simulation.

Simulation of the GWL algorithm involves a number of stages. Let $\mathbf{x}^{(s,k)}$ denote a sample drawn at iteration k of stage s, and let $\hat{g}^{(s,k)}(E_i)$ denote the estimate of $g(E_i)$ at iteration k of stage s. In the first stage ($s = 1$ and $k = 0$), the simulation starts with the initial estimates $\hat{g}^{(1,0)}(E_1) = \cdots = \hat{g}^{(1,0)}(E_m) = 1$ and a random sample $\mathbf{x}^{(1,0)}$ drawn from $\mathcal{X}_0$ according to an arbitrary distribution, and then iterates between the following two steps:

a. Sampling: Propose a new configuration $\mathbf{x}^*$ according to the proposal distribution $T(\mathbf{x}^*|\mathbf{x}^{(s,k)})$. Accept $\mathbf{x}^*$ with probability

$$\min\left\{\frac{\hat{g}^{(s,k)}(E_{I_{\mathbf{x}^{(s,k)}}})}{\hat{g}^{(s,k)}(E_{I_{\mathbf{x}^*}})} \frac{\psi(\mathbf{x}^*)}{\psi(\mathbf{x}^{(s,k)})} \frac{T(\mathbf{x}^{(s,k)}|\mathbf{x}^*)}{T(\mathbf{x}^*|\mathbf{x}^{(s,k)})}, 1\right\}, \tag{7}$$

where $I_{\mathbf{z}}$ denotes the index of the subregion to which $\mathbf{z}$ belongs. If it is accepted, then set $\mathbf{x}^{(s,k+1)} = \mathbf{x}^*$; otherwise, set $\mathbf{x}^{(s,k+1)} = \mathbf{x}^{(s,k)}$.

b. Weight updating: Set

$$\hat{g}^{(s,k+1)}\big(E_{I_{\mathbf{x}^{(s,k+1)}}+i}\big) = \hat{g}^{(s,k)}\big(E_{I_{\mathbf{x}^{(s,k+1)}}+i}\big) + \delta_s \rho^i \hat{g}^{(s,k)}\big(E_{I_{\mathbf{x}^{(s,k+1)}}}\big) \qquad \text{for } i = 0, \ldots, m - I_{\mathbf{x}^{(s,k+1)}}.$$

The iteration continues until a stable histogram (described later) has been produced in the space of subregions. Once the histogram is stable, the simulation will proceed to the next stage, resuming the sample collector, reducing the modification factor to a smaller value, and passing on the current estimate and sample as initial values to the new stage; that is, setting $\mathbf{x}^{(s,0)} = \mathbf{x}^{(s-1,K_{s-1})}$ and $\hat{g}^{(s,0)}(E_i) = \hat{g}^{(s-1,K_{s-1})}(E_i)$ for $i = 1, \dots, m$, where K_{s-1} denotes the total number of iterations performed in stage $s-1$. The process is repeated until $\delta_s < \delta_{\text{end}}$, where δ_{end} is usually a very small number, say 10^{-7}.

In the GWL algorithm $\mathbf{x}^{(s,0)} = \mathbf{x}^{(s-1,K_{s-1})}$ is set for convenience only. Theoretically, $\mathbf{x}^{s,0}$ can be set to any random sample drawn from $\mathcal{X}_0$ as $\mathbf{x}^{(1,0)}$. The modification factor δ_1 is usually set to a large number, say 1 or 2, which allows the sampler to reach all subregions very quickly even for a large system. It will then decrease strictly in the following stage. For example, it can decrease in a scheme like $\delta_s = \gamma\delta_{s-1}$ with $\gamma < 1$, or $\delta_s = \sqrt{1+\delta_{s-1}} - 1$, as suggested by Wang and Landau (2001). The latter was used in all examples of this article with $\delta_1 = e - 1 = 1.718$.

A histogram is considered stable if its shape will not change much with further samples generated from the same process. In this article we define the following statistic to measure the stability of the histogram:

$$S(k,b) = \frac{1}{m}\sum_{i=1}^{m}\left|\frac{\hat{P}_{i,(k+1)b}}{\hat{P}_{i,kb}} - 1\right|, \tag{8}$$

where b is the batch size and $\hat{P}_{i,kb}$ is the normalized sampling frequency of subregion E_i within the first kb iterations of the stage. In S_k, $0/0 = 1$ is defined to accommodate the empty E_i's. Because $S_k \to 0$ as $k \to \infty$, a threshold value can be chosen for S_k to monitor the stability of the histogram. Our experience shows that .01 may be a good choice for the threshold value.

At each iteration, the sampling step can be repeated a number of times; that is, generating $\mathbf{x}^{(s,k+1)}$ with multiple MH steps instead of only one step. The repetition will improve the approximation accuracy of $\hat{g}$ to g, but should not affect its convergence even for a very large system (Wang and Landau 2001). The repeat also renders a rigorous theoretical analysis for the algorithm by treating $\mathbf{x}^{(s,k)}$ as a sample drawn from the trial density $\pi^{(s,k)}(\mathbf{x}) \propto \sum_{i=1}^{m} \psi(\mathbf{x})/\hat{g}^{(s,k)}(E_i) I(\mathbf{x} \in E_i)$, $\mathbf{x} \in \mathcal{X}$. The following theorem is shown based on the assumption that $\mathbf{x}^{(s,k)}$ is a sample drawn from $\pi^{(s,k)}(\mathbf{x})$. Because number of sampling steps performed in each iteration will not affect the convergence of the algorithm, and for simplicity, the sampling step is performed only once in all simulations of this article.

Theorem 1. Suppose that the GWL algorithm satisfies the following conditions: (A_1) $\psi(\mathbf{x})$ is a nonnegative function defined on $\mathcal{X}$ with $0 < \int_{\mathcal{X}} \psi(\mathbf{x})\,d\mathbf{x} < \infty$; ($\text{A}_2$) the proposal distribution is global; and (A_3) $\delta_s \to 0$ as $s \to \infty$. As $s \to \infty$ and $k \to \infty$, we have

$$\hat{g}^{(s,k)}(E_i) \longrightarrow cg(E_i) \quad \text{in probability}, \tag{9}$$

for $i = 1, \dots, m$, where $g(E_i)$'s are defined in (6) and c is a constant. The value of c can be determined by imposing an additional constraint on the $\hat{g}(E_i)$'s, for example, $\sum_{i=1}^{m} \hat{g}(E_i) = \sum_{i=1}^{n} g(E_i)$ if $\sum_{i=1}^{m} g(E_i)$ is known or $\hat{g}(E_j) = g(E_j)$ if $g(E_j)$ is known for some j.

As a byproduct of the proof of Theorem 1, we have the following corollary, which gives the approximation accuracy of $\hat{g}(E_i)$ to $g(E_i)$ at each stage.

Corollary 1. At each stage s, we have

$$\sum_{i=1}^{m} E\big[\hat{g}(E_i)^{(s,k)} - g(E_i)\big]^2 \approx O(\delta_s),$$

as $k \to \infty$.

If the $\hat{g}(E_i)$'s have converged, then the visiting frequency to each subregion must be proportional to

$$\frac{\int_{E_i} \psi(\mathbf{x})\,d\mathbf{x}}{g(E_i)} \quad \text{for } i = 1, \dots, m, \tag{10}$$

because of the acceptance rule (7). Here $0/0 = 0$ is defined to accommodate the case $g(E_i) = 0$. If $\rho = 0$, then (10) implies that each of the subregions with $g(E_i) > 0$ will be sampled from equally. In this sense, we say that GWL leads to a free random walk in the space of subregions while within the same subregion, GWL is reduced to the conventional MH algorithm. If $\rho > 0$, then GWL will also lead to a random walk in the space of subregions, but with more weight toward low-indexed regions. If we code the subregions appropriately such that the subregion from which we want to sample most frequently is coded as E_1, and then E_2, E_3, and so on, then a choice of $\rho > 0$ will often result in an efficient simulation. This typically happens in Monte Carlo optimization, where one needs to bias sampling to low-energy regions. In what follows, we sometimes denote the GWL algorithm by ρ-GWL to emphasize that ρ is taking a nonzero value.

The ρ-GWL algorithm is so general that it has included several other algorithms as its special cases. If $\psi(\mathbf{x}) \equiv 1$, $\mathcal{X}$ is finite, $T(\cdot|\cdot)$ is a symmetric function, and $\rho = 0$, then the algorithm is reduced to the WL algorithm. In this case $\hat{g}$ estimates the spectral density of the system. If $\psi(\mathbf{x}) \equiv 1$, $\mathcal{X}$ is finite, and $\rho = 1$, then the algorithm gives a WL-style implementation for $1/k$-ensemble sampling. In this case $\hat{g}$ estimates the cumulative spectral density of the system.

4. AN ILLUSTRATIVE EXAMPLE

In this section the GWL algorithm is illustrated by and compared with the multicanonical algorithm through an example. The distribution consists of 10 states with the unnormalized mass function $P(x)$ as specified in Table 1. The transition matrix used is a random matrix of which each row is generated independently from Dirichlet$(1, \dots, 1)$. The proposal distribution is global.

The GWL algorithm was applied to this example. The state space was partitioned according to the unnormalized mass function into seven subregions: $E_1 = \{x : \log P(x) > 5\} = \{8\}$, $E_2 = \{x : 5 \ge \log P(x) > 4\} = \{2\}$, $E_3 = \{x : 4 \ge \log P(x) > 3\} = \emptyset$, $E_4 = \{x : 3 \ge \log P(x) > 2\} = \emptyset$, $E_5 = \{x : 2 \ge \log P(x) > 1\} = \{5, 6\}$, $E_6 = \{x : 1 \ge \log P(x) > 0\} = \{3, 9\}$, and $E_7 = \{x : \log P(x) \le 0\} = \{1, 4, 7, 10\}$. The GWL algorithm was run 20 times independently. In the first 10 runs, $\psi(x) = P(x)$, $\rho = 0$, $\delta_{\text{end}} = 10^{-6}$, $n_1 = 10{,}000$, and $n_{s+1} = 1.2n_s$, where n_s denotes the number of iterations of stage s. The total CPU time cost for

Table 1. The Unnormalized Mass Function of the 10-State Distribution

x	1	2	3	4	5	6	7	8	9	10
$P(x)$	1	100	2	1	3	3	1	200	2	1

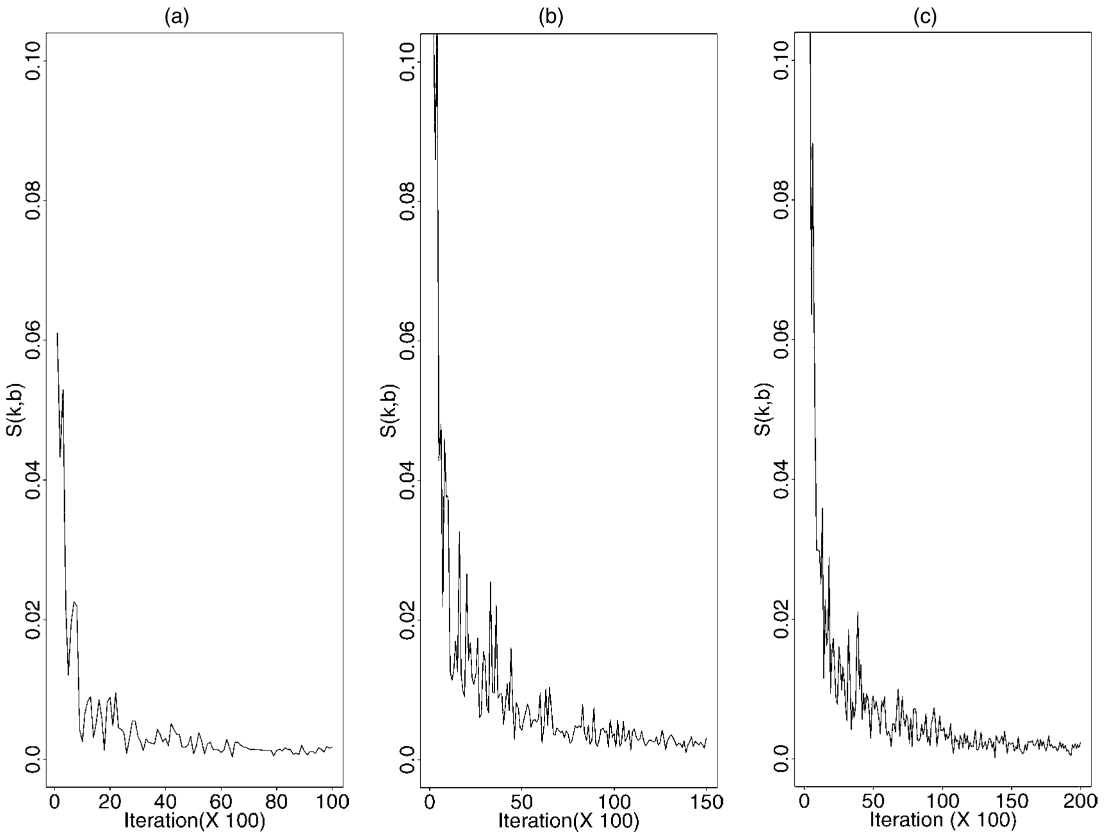

Figure 1. Curves of $S(k, b)$ ($b = 100$) for the Histograms Obtained in Stages (a) 1, (b) 10, and (c) 20 of a Run.

the 10 runs was about 4 seconds on a 2.8-GHz computer. (All computations reported in this article were done on the same computer.) Figure 1 examines the stability of the histograms produced in stages 1, 10, and 20 of a run by plotting the curves of $S(k, b)$. The plots indicate that the GWL algorithm converges slower and slower as δ_s decreases. This suggests that a larger number of iterations should be used in the latter stages of the simulation. In the next 10 runs, the GWL algorithm had the same settings as used in the first 10 runs except for $\rho = 1$. The computational results are summarized in Table 2. In the GWL algorithm, the constraint $\sum_{i=1}^{7} \hat{g}(E_i) - \rho\hat{g}(E_i) = 314$ [setting $g(E_0) = 0$] is imposed on $\hat{g}$ to remove the unknown constant c of Theorem 1.

For the foregoing partition, it happens that each subregion is either empty or contains only states with the same energy values. Hence the multicanonical and WL algorithms can still be used to estimate the spectral density of the distribution. The WL algorithm was run for 10 times independently with the same setting as that used for the GWL algorithm except for $\psi(x)$ and the transition matrix. In the WL algorithm $\psi(x) = 1$, and the

Table 2. Computational Results for the 10-State Distribution

				Multicanonical		Wang–Landau		GWL ($\rho = 0$)		ρ-GWL ($\rho = 1$)	
Subregion	State	Ω	$g(E)$	$\hat{\Omega}$	$F_{(\times 10^{-4})}$	$\hat{\Omega}$	$F_{(\times 10^{-4})}$	$\hat{g}(E)$	$F_{(\times 10^{-4})}$	$\hat{g}(E)$	$F_{(\times 10^{-4})}$
E_1	8	1	200	$1.002_{(.007)}$	$.20_{(15.74)}$	$1.001_{(.001)}$	$.20_{(3.24)}$	$199.93_{(.11)}$	$.20_{(3.61)}$	$200.13_{(.10)}$	$.725_{(5.81)}$
E_2	2	1	100	$.995_{(.003)}$	$.20_{(13.57)}$	$1.000_{(.001)}$	$.20_{(4.24)}$	$100.07_{(.11)}$	$.20_{(3.47)}$	$99.88_{(.11)}$	$.242_{(6.49)}$
E_3		0	0	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$
E_4		0	0	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$	$0_{(0)}$
E_5	5	2	6	$1.996_{(.010)}$	$.10_{(7.75)}$	$2.000_{(.001)}$	$.10_{(1.25)}$	$6.00_{(.01)}$	$.10_{(3.24)}$	$5.99_{(.02)}$	$.007_{(.58)}$
	6				$.10_{(5.33)}$		$.10_{(1.96)}$		$.10_{(2.24)}$		$.007_{(1.08)}$
E_6	3	2	4	$2.002_{(.008)}$	$.10_{(9.67)}$	$2.000_{(.002)}$	$.10_{(1.90)}$	$4.00_{(.01)}$	$.10_{(3.32)}$	$3.96_{(.02)}$	$.005_{(.72)}$
	9				$.10_{(8.20)}$		$.10_{(2.12)}$		$.10_{(2.95)}$		$.005_{(.20)}$
E_7	1	4	4	$4.005_{(.016)}$	$.05_{(4.11)}$	$3.999_{(.002)}$	$.05_{(.98)}$	$4.00_{(.01)}$	$.05_{(1.52)}$	$4.03_{(.04)}$	$.002_{(.20)}$
	4				$.05_{(3.28)}$		$.05_{(1.13)}$		$.05_{(1.93)}$		$.002_{(.62)}$
	7				$.05_{(2.93)}$		$.05_{(1.23)}$		$.05_{(2.73)}$		$.002_{(.47)}$
	10				$.05_{(2.92)}$		$.05_{(1.64)}$		$.05_{(1.43)}$		$.002_{(.33)}$

NOTE: The numbers before and in the parentheses are the average of the estimates and the standard deviation of the average value. These numbers are calculated with 10 independent runs. The "F" column records the average of the sampling frequency of each state and the standard deviation of the average value. The standard deviation is reported on a scale of 10^{-4}.

transition matrix is a constant matrix with all elements equal to .1. The multicanonical algorithm was also run for 10 times independently. Each run consisted of 21 stages, including the initial stage. Each stage consisted of 1.1×10^5 iterations. The total CPU time for those 10 runs was also about 4.0 seconds. The computation results are summarized in Table 2. In the two algorithms, the constraint $\sum_{i=1}^{7} \hat{\Omega}(u_i) = 10$ is imposed on $\hat{\Omega}$ for obtaining a valid estimate for the spectral density.

The multicanonical and WL algorithms estimate the spectral density of the distribution, whereas the GWL algorithm estimates the weights defined for each of the subregions. The multicanonical, WL, and GWL (with $\rho = 0$) algorithms all sample from each of the subregions equally, regardless of the true spectral density or the weight of each subregion. This is not the case for the ρ-GWL algorithm ($\rho > 0$). The ρ-GWL algorithm tends to sample more frequently from the low-indexed subregions. Table 2 shows that the plain GWL algorithm ($\rho = 0$) may be better than the ρ-GWL algorithm if our target is to estimate g. We explore this point theoretically in Section 7. As a byproduct, Table 2 also shows that the WL algorithm produces much more accurate estimates than the multicanonical algorithm for this example.

5. USE OF THE GENERALIZED WANG–LANDAU ALGORITHM IN MONTE CARLO INTEGRATION

As mentioned in Section 1, the GWL algorithm provides a new approach for estimating the expectation $E_f h(\mathbf{x})$. However, Theorem 1 demonstrates that the expectation can be approximated by the GWL algorithm only up to an unknown constant c if there is no extra information available on the $g(E_i)$'s. To determine the value of c, we use two tricks when there is no extra information available on the $g(E_i)$'s. First, truncate $\mathcal{X}$ to a bounded region $\widetilde{\mathcal{X}}$ if $\mathcal{X}$ is unbounded. This can be done as follows. Run the GWL algorithm on the space $\mathcal{X}$ with an appropriate partition, and identify a bounded region $\widetilde{\mathcal{X}}$ such that the ratio $\int_{\widetilde{\mathcal{X}}^c} h(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x} / \int_{\mathcal{X}} h(\mathbf{x}) f(\mathbf{x})\, d\mathbf{x}$ is negligible, say $< .001$, where $\widetilde{\mathcal{X}}^c = \mathcal{X} - \widetilde{\mathcal{X}}$ is the complementary set of $\widetilde{\mathcal{X}}$. Note that the ratio can be estimated with the $\hat{g}(E_i)$'s, as shown in Section 5.2. This truncation is different from the conventional distribution truncation. It is made on the integral instead of the density function $f(\mathbf{x})$, and the resulting region $\widetilde{\mathcal{X}}$ may be different for a different $h(\mathbf{x})$. Hence it may not be so harmful to the latter statistical inference as the conventional distribution truncation. Kass and Wasserman (1996) gave an overview on the influence of distribution truncation. Second, augment $\widetilde{X}$ by adding a small region $\mathcal{A}$ disjoint from $\widetilde{\mathcal{X}}$, and specify a function on $\mathcal{A}$ such that the integral $\int_{\mathcal{A}} \psi(\mathbf{x})\, d\mathbf{x}$ is tractable analytically. With the foregoing tricks, the expectation $E_f h(\mathbf{x})$ can be estimated as follows, providing that $h(\mathbf{x}) \ge 0$ and $0 < E_f |h(\mathbf{x})| < \infty$:

a. Define

$$f^*(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \mathbf{x} \in \widetilde{\mathcal{X}} \\ \dfrac{1}{a}, & \mathbf{x} \in \mathcal{A}, \end{cases} \qquad \text{and}$$

$$h^*(\mathbf{x}) = \begin{cases} h(\mathbf{x}), & \mathbf{x} \in \widetilde{\mathcal{X}} \\ 1, & \mathbf{x} \in \mathcal{A}, \end{cases}$$

where $a = \int_{\mathcal{A}} d\mathbf{x}$ is the hypervolume of the region $\mathcal{A}$.

b. Code region $\mathcal{A}$ as E_0, and partition the sample space $\widetilde{\mathcal{X}}$ into m disjoint subregions, $E_1, \ldots, E_m$, according to a chosen criterion.

c. Set $\psi(\mathbf{x}) = h^*(\mathbf{x}) f^*(\mathbf{x})$, and run the GWL algorithm on the space $\widetilde{\mathcal{X}} \cup \mathcal{A}$.

d. Estimate the integral $E_f h(\mathbf{x})$ by

$$\sum_{i=1}^{m} [\hat{g}(E_i) - \rho \hat{g}(E_{i-1})] / \hat{g}(E_0). \tag{11}$$

Because $g(E_0) > 0$, it follows from Theorem 1 that estimate (11) is consistent as $s \to \infty$ and $k \to \infty$. For a general function $h(\mathbf{x})$, the expectation can be calculated by considering two integrals, $E_f h^+(\mathbf{x})$ and $E_f h^-(\mathbf{x})$, where $h^+(\mathbf{x})$ and $h^-(\mathbf{x})$ are defined as

$$h^+(\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{if } h(\mathbf{x}) > 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{and}$$

$$h^-(\mathbf{x}) = \begin{cases} -h(\mathbf{x}) & \text{if } h(\mathbf{x}) < 0 \\ 0 & \text{otherwise.} \end{cases}$$

For simplicity, we assume that $\mathcal{X}$ is bounded in the following examples of this section. Thus we have $\mathcal{X} = \widetilde{\mathcal{X}}$.

5.1 A Demonstration Example

We use this example to demonstrate the validity of the GWL algorithm for Monte Carlo integration. The distribution is a mixture of truncated bivariate normal distributions. The density function is

$$f(\mathbf{x}) = \frac{1}{Z}\left[\frac{1}{3}\exp\left\{-\frac{1}{2}\sum_{i=1}^{2}(x_i+5)^2\right\} + \frac{2}{3}\exp\left\{-\frac{1}{2}\sum_{i=1}^{2}(x_i-5)^2\right\}\right] I(\mathbf{x} \in \mathcal{X}),$$

where $\mathcal{X} = [-10, 10]^2$, $I(\cdot)$ is the indicator function, and Z is the normalizing constant. In the first experiment, Z is estimated using the GWL algorithm. The true value of Z is 2π. The sample space $\mathcal{X}$ is augmented by $\mathcal{A} = [10, 11] \times [-10, 10]$, and $\psi(\mathbf{x})$ is set as

$$\psi(\mathbf{x}) = \begin{cases} \dfrac{1}{3}\exp\left\{-\dfrac{1}{2}\sum_{i=1}^{2}(x_i+5)^2\right\} \\ \quad + \dfrac{2}{3}\exp\left\{-\dfrac{1}{2}\sum_{i=1}^{2}(x_i-5)^2\right\}, & \mathbf{x} \in \mathcal{X} \\ .05, & \mathbf{x} \in \mathcal{A}. \end{cases}$$

The region $\mathcal{A}$ is coded as E_0, and $\mathcal{X}$ is partitioned into the following subregions: $E_1 = \{\mathbf{x} \in \mathcal{X} : -\log\psi(\mathbf{x}) \le -4.9\}$, $E_2 = \{\mathbf{x} \in \mathcal{X} : -4.9 < -\log\psi(\mathbf{x}) \le -4.8\}, \ldots,$ and $E_{351} = \{\mathbf{x} \in \mathcal{X} : 30 < -\log\psi(\mathbf{x})\}$. In simulation, the following are set: $\rho = 0$, the proposal distribution $\mathrm{N}_2(\mathbf{x}_k, 3^2\mathbf{I}_2)$, $\delta_{\text{end}} = 10^{-7}$, $n_1 = 10^5$, and $n_{s+1} = 1.1 n_s$. Here a small variance is set for the proposal distribution to test the ability of the GWL algorithm in escaping from local energy minima. A run of the GWL algorithm costs about 10 seconds of CPU time. The overall acceptance rate of the GWL moves is about .6. The computational results are summarized in Figures 2(a) and 2(b). Figure 2(a) shows the convergence of the estimate of Z, where the y-axis is defined as $\mathit{Distance} = |\hat{Z}_s - 6.28|/6.28$, with $\hat{Z}_s$ being the estimate of Z obtained at stage s. Figure 2(b) shows the sampling frequencies of the subregions $E_0, \ldots, E_{351}$ at the last stage. Because $-\log\psi(\mathbf{x}) > .4$ for all $\mathbf{x} \in \mathcal{X}$, the sets $E_1, \ldots, E_{54}$ are all empty and the corresponding sampling frequencies are 0. Figure 2(b) indicates that the GWL algorithm ($\rho = 0$) samples from each of the nonempty subregions equally. Later, the run was repeated for 50 times independently. By averaging over the 50 runs, we obtained one estimate for Z, 6.26 with standard deviation .05.

In the second experiment, $E_f X_1^2$ is estimated using the GWL algorithm. The true value of the expectation is 26.0. The GWL algorithm was run for this example with the same setting as that used in the first experiment except for $\psi(\mathbf{x})$, which was

$$\psi(\mathbf{x}) = \begin{cases} \dfrac{x_1^2}{2\pi}\left[\dfrac{1}{3}\exp\left\{-\dfrac{1}{2}\sum_{i=1}^{2}(x_i+5)^2\right\}\right. \\ \qquad \left. + \dfrac{2}{3}\exp\left\{-\dfrac{1}{2}\sum_{i=1}^{2}(x_i-5)^2\right\}\right], & \mathbf{x} \in \mathcal{X}, \\ .05, & \mathbf{x} \in \mathcal{A}. \end{cases}$$

The computational results are summarized in Figures 2(c) and 2(d). Figure 2(c) shows the convergence of the estimate of $E_f X_1^2$, where the y-axis is defined as $\mathit{Distance} = |\hat{\xi}_s - 26|/26$ with $\hat{\xi}_s$ being the estimate of $E_f X_1^2$ obtained at stage s. Figure 2(d) shows the sampling frequencies of the subregions $E_0, \ldots, E_{351}$ at the last stage. Because $-\log\psi(\mathbf{x}) > -1.2$, $E_1, \ldots, E_{39}$ are all empty, and the corresponding sampling frequencies are all 0. Later the run was repeated 50 times independently. By averaging over the 50 runs, we obtained one estimate for $E_f X_1^2$, 25.8 with standard deviation .19.

We note that the tasks studied in this section can also be accomplished by other Monte Carlo methods. For example, the normalizing constant Z can be estimated by the methods presented by Meng and Wong (1996) or Chen, Shao, and Ibrahim (2000). Here we just provide a different treatment for the problems and do not want to compare their efficiency, because the strength of the GWL algorithm is in overcoming the high-energy barriers of a rough energy landscape, whereas this example is still relatively simple.

5.2 Computing Highest Posterior Density Intervals

Consider a Bayesian posterior density $f(\theta, \boldsymbol{\phi}|D)$, where D denotes data, θ is a one-dimensional parameter, and $\boldsymbol{\phi}$ may be a multidimensional vector of parameters other than θ in the model. A $100(1-\alpha)\%$ HPD interval for θ is given by

$$R(f_\alpha) = \{\theta : f(\theta|D) \geq f_\alpha\},$$

where f_α is the largest constant such that $P(\theta \in R(f_\alpha)) \geq 1 - \alpha$. The HPD interval has the following two properties:

a. Every point inside the interval has greater density than every point outside the interval.

b. For a given probability content, say $1 - \alpha$, the interval is of the shortest length.

HPD intervals are usually used to delineate a comparatively small set that contains most of the probability in summarizing a distribution but may be nonzero over infinite regions of the sample space. When $f(\theta|D)$ is multimodal, $R(f_\alpha)$ is actually a collection of intervals and should be called a $100(1-\alpha)\%$ HPD region. In this article we follow Chen et al. (2000) to still call it a HPD interval, considering that θ is one-dimensional.

Let $F(\theta|D)$ denote the cdf of $f(\theta|D)$. When the closed forms of $f(\theta|D)$ and $F(\theta|D)$ are available and $f(\theta|D)$ is continuous

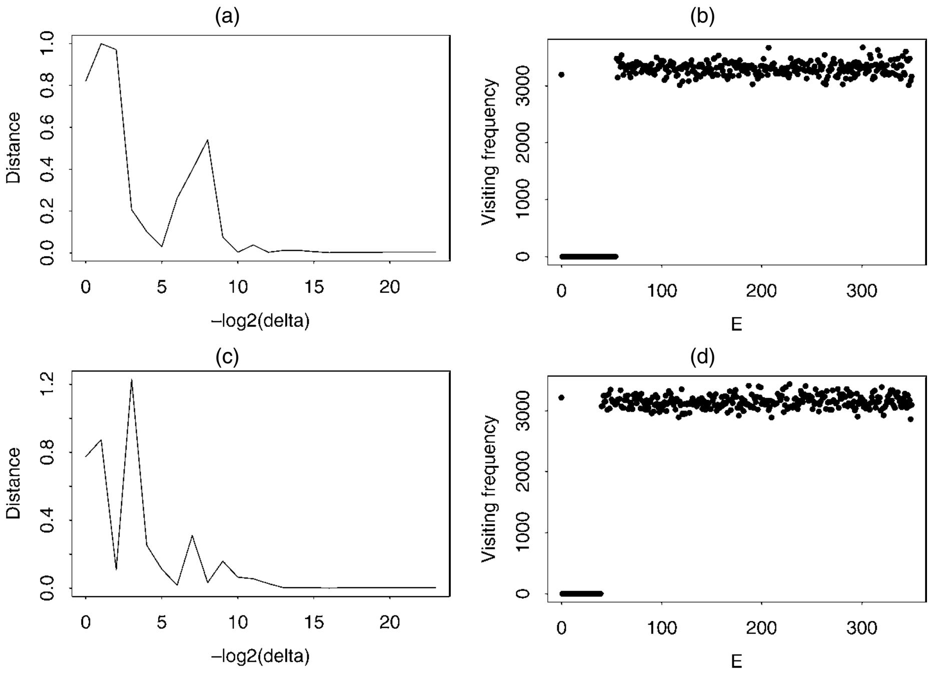

Figure 2. Results of the First Experiment [(a) and (b)] and the Second Experiment [(c) and (d)]. (a) The convergence of the estimate of the normalizing constant. (b) The sampling frequencies of the subregions, $E_0, \ldots, E_{351}$, at the last stage. (c) The convergence of the estimate of $E_f X_1^2$. (d) The sampling frequencies of the subregions, $E_0, \ldots, E_{351}$, at the last stage.

and unimodal, $R(f_\alpha)$ can be computed by solving the following minimization problem:

$$\min_{\theta_L<\theta_U} \left\{ \left|f(\theta_U|D) - f(\theta_L|D)\right| + \left|F(\theta_U|D) - F(\theta_L|D) - (1-\alpha)\right| \right\},$$

where θ_L and θ_U are the lower and upper bounds of the HPD interval. However, in Bayesian computation the closed forms of $f(\theta|D)$ and $F(\theta|D)$ are usually not available, especially when $\boldsymbol{\phi}$ is high-dimensional. Various Monte Carlo methods have been proposed to approximate $R(f_\alpha)$ (see, e.g., Wei and Tanner 1990; Tanner 1996; Hyndman 1996; Chen and Shao 1999). The method developed by Wei and Tanner (1990), Tanner (1996), and Hyndman (1996) approximates f_α as follows. Let $\theta_1, \dots, \theta_n$ denote random samples drawn from $f(\theta|D)$, and let $\xi_i = f(\theta_i|D)$ for $i = 1, \dots, n$. A consistent estimator of $R(f_\alpha)$ can then be obtained by computing

$$\hat{R}(f_\alpha) = \{\theta : f(\theta|D) \ge \hat{f}_\alpha\},$$

where $\hat{f}_\alpha = \xi_{[n\alpha]}$, the $[n\alpha]$th smallest of the $\{\xi_i\}$'s. This method has two drawbacks. First, it requires that a closed form of $f(\theta|D)$ be available; second, it is difficult to compute $\hat{R}(f_\alpha)$, especially when $f(\theta|D)$ is multimodal. Chen and Shao (1999) proposed a method to estimate $R(f_\alpha)$ based on the order statistics of the random samples. Although their method avoids the requirement for the closed form of $f(\theta|D)$, it requires that $f(\theta|D)$ be unimodal. Extension to the multimodal case is difficult.

Later we give a method for estimating $R(f_\alpha)$ using the GWL algorithm. Comparing to the methods reviewed earlier, our method has the following advantages: It avoids the requirement for the closed form of $f(\theta|D)$ and is valid for the multimodal case. Suppose that $f(\theta|D)$ is continuous and has a moderate number of modes, and that $f(\theta|D)$ is defined on the interval $\Theta = (\theta_{\min}, \theta_{\max})$ or that the total probability outside this interval is negligible. Furthermore, suppose that the sample space has been partitioned into the subregions $E_1 = \{(\theta, \boldsymbol{\phi}) : \theta_{\min} < \theta \le \theta_1\}$, $E_2 = \{(\theta, \boldsymbol{\phi}) : \theta_1 < \theta \le \theta_2\}, \dots,$ and $E_m = \{(\theta, \boldsymbol{\phi}) : \theta_{m-1} < \theta \le \theta_{\max}\}$, where $\theta_{\min}, \theta_1, \dots, \theta_{m-1}, \theta_{\max}$ are equally spaced. Here m is usually a large number that is at least several folds of the number of the modes of $f(\theta|D)$. Let $e_i = \{\theta : \theta_1 < \theta \le \theta_i\}$ be a subinterval of θ corresponding to E_i, $i = 1, \dots, m$. The HPD interval can be constructed as follows.

Procedure I.

a. Run the GWL algorithm with $\psi(\theta, \boldsymbol{\phi}) = f(\theta, \boldsymbol{\phi}|D)$.

b. Compute

$$P(E_i) = \frac{\hat{g}(E_i) - \rho\hat{g}(E_{i-1})}{\sum_{j=1}^m [\hat{g}(E_j) - \rho\hat{g}(E_{j-1})]}, \qquad i = 1, \dots, m, \tag{12}$$

where $g(E_0) = 0$. It follows from Theorem 1 that $P(E_i)$ forms a consistent estimator for the quantity $\int_{E_i}\int_\Phi f(\theta, \boldsymbol{\phi}|D)\, d\theta\, d\boldsymbol{\phi}$, where Φ denotes the sample space of $\boldsymbol{\phi}$.

c. Order the subregions in descending order by $P(E_i)$'s. Let $E_{(1)}, \dots, E_{(m)}$ denote the ordered subregions, and let $e_{(1)}, \dots, e_{(m)}$ be the corresponding subintervals. Construct the HPD interval as

$$\hat{R}(f_\alpha) = \bigcup_{i=1}^{m_0} e_{(i)}, \tag{13}$$

where $m_0 = \min\{k : \sum_{i=1}^k P(E_{(i)}) \ge 1 - \alpha\}$.

d. If $\sum_{i=1}^{m_0} P(E_{(i)}) \le 1 - \alpha + \epsilon$, stop; otherwise, refine the partition and go to step a. The ϵ is the tolerance error specified by the user.

The accuracy of $\hat{R}(f_\alpha)$ depends on the length of the subintervals, and it can be improved by refining the partition. By the fundamental theorem of calculus, we know that

$$\sum_{i=1}^{m_0} P\big(E_{(i)}\big) \to \int_{\bigcup_{i=1}^{m_0} E_{(i)}} f(\theta|D)\, d\theta,$$

as $m \to \infty$ or, equivalently, the length of the subinterval tends to 0. This implies that $\hat{R}(f_\alpha)$ forms a consistent estimator for $R(f_\alpha)$. In practice, a reasonable criterion for the sample space partition is to partition the sample space such that $P(E_{(1)}) < \zeta$, where ζ is a user set tolerance value. Augmentation of a region $\mathcal{A}$ to $\mathcal{X}$ is not needed in the foregoing procedure, as we are interested only in the ratios of the $[\hat{g}(E_i) - \rho\hat{g}(E_{i-1})]$'s instead of their values.

When the closed form of $f(\theta|D)$ is available, Procedure I can be refined as follows.

Procedure II.

a. Run the GWL algorithm with $\psi(\theta) = f(\theta|D)$ and the current sample space partition, $E_1, \dots, E_m$. Let $\theta_1, \dots, \theta_n$ denote the samples drawn by the GWL algorithm in the run.

b. Compute

$$P(E_i) = \frac{\hat{g}(E_i) - \rho\hat{g}(E_{i-1})}{\sum_{j=1}^m [\hat{g}(E_j) - \rho\hat{g}(E_{j-1})]}, \qquad i = 1, \dots, m, \tag{14}$$

where $g(E_0) = 0$.

c. Construct the HPD interval as

$$\hat{R}(f_\alpha) = \bigcup_{i=1}^{m_1} E_i^*, \tag{15}$$

where $m_1 = \max\{k : \sum_{i=1}^k P(E_i^*) \le 1 - \alpha + \epsilon_1, \inf_{\bigcup_{i=1}^k E_i^*} f(\theta|D) \ge \sup_{\bigcup_{i=1}^{m-k} E_i^+} f(\theta|D)\}$. Here the E_i^*'s and E_i^+'s denote the subintervals included in and excluded from (15). If $\sum_{i=1}^{m_1} P(E_i^*) \ge 1 - \alpha - \epsilon_2$, then stop; otherwise, go to step d. The ϵ_1 and ϵ_2 are the tolerance errors chosen by the user.

d. Construct the HPD interval as

$$\hat{R}(f_\alpha) = \left(\bigcup_{i=1}^{m_1} E_i^*\right) \cup \left(\bigcup_{i=1}^{m_2} E_i^{**}\right), \tag{16}$$

where $m_2 = \min\{k : \sum_{i=1}^k P(E_i^*) + \sum_{i=1}^k P(E_i^{**}) \ge 1 - \alpha, \inf_{\bigcup_{i=1}^k E_i^{**}} f(\theta|D) \ge \sup_{\bigcup_{i=1}^{m-m_1-k} E_i^{++}} f(\theta|D)\}$. Here the E_i^{**}'s denote the subintervals selected from the set $\{E_i^+ : i = 1, \dots, m - m_1\}$, and the E_i^{++}'s denote the unselected subintervals by (15) and (16).

e. If a positive m_2 is found in step d, refine the subintervals $E_1^{**}, \dots, E_{m_2}^{**}$, reset the value of m, and go to step a. Otherwise, refine the subintervals $E_1^+, \dots, E_{m-m_1}^+$, reset the value of m, and go to step a.

In the foregoing procedure, $\inf_A f(\theta|D)$ and $\sup_A f(\theta|D)$ can be estimated by $\min_{\theta_k \in A} f(\theta_k|D)$ and $\max_{\theta_k \in A} f(\theta_k|D)$, where the minimization and maximization are taken over all GWL samples $\theta_1, \dots, \theta_n$. These estimators are consistent. Compared with Procedure I, a smaller value of m is often used here as an award for the availability of the closed form of $f(\theta|D)$.

In what follows we consider one example for which the density function is

$$f(\theta,\phi) = \frac{1}{Z}\left[\frac{1}{3}\exp\left\{-\frac{1}{2}\big((\theta+5)^2+(\phi+5)^2\big)\right\} + \frac{2}{3}\exp\left\{-\frac{1}{2}\big((\theta-5)^2+(\phi-5)^2\big)\right\}\right] \times I\big(\theta\in(-10,10)\big),$$

which mimics a multimodal posterior density.

Procedure I was applied to this example by assuming that $f(\theta)$ is not available analytically. The GWL algorithm was run for 10 times independently with $\rho = 0$, $\Theta = (-10, 10)$, $m = 500$, $\delta_{\text{end}} = 10^{-7}$, $n_1 = 5 \times 10^5$, $n_{s+1} = 1.1n_s$, and the proposal density $\mathrm{N}(\mathbf{x}_k, 3^2\mathbf{I}_2)$. The CPU time cost for each run was about 52 seconds. The overall acceptance rate was about .29. The computational results are summarized in Figure 3 and show that the GWL algorithm produced a rather accurate estimate for the marginal density of θ in each of the 10 runs. By averaging over the 10 runs, we obtained one estimate for the weight of the left component of $f(\theta, \boldsymbol{\phi})$ and one estimate for the 90% HPD interval. The estimate of the weight is .3337 with standard deviation .0021, which is almost identical to the true value 1/3. The estimate of the interval is $(-6.35, 3.61) \cup (3.20, 6.83)$. The standard deviations of the four endpoints are .008, .005, .004, and .005. The coverage probability of the interval is .897. Comparing this with the true 90% HPD interval $(-6.4, -3.6) \cup (3.17, 6.82)$, we see that the GWL algorithm has produced a quite accurate estimate for the HPD interval for this example. We have tried other settings, for example, $m = 200$ and $n_{i+1} = 1.2n_i$; the results are similar.

6. USE OF THE GENERALIZED WANG–LANDAU ALGORITHM IN MONTE CARLO OPTIMIZATION

Suppose that the energy function $H(\mathbf{x})$ has a very rough landscape, with many local minima separated by high-energy barriers. As mentioned in Section 1, $H(\mathbf{x})$ can be minimized by simulating from the distribution

$$f(\mathbf{x}) \propto \exp\left\{-\frac{H(\mathbf{x})}{t}\right\} \tag{17}$$

with a Monte Carlo algorithm, where t is the temperature. Typical algorithms include simulated annealing, simulated tempering, and parallel tempering. These algorithms use the same technique, trying to escape from local energy minima by simulating at high temperatures. This raises the question of how high a temperature is suitable for a given energy function. If the temperature is too high, then simulating from (17) will be almost equivalent to making a random walk in the space $\mathcal{X}$. Hence location the attraction basin of the global minimum will be very difficult, especially when $\mathcal{X}$ is large, whereas if the temperature is too low, then locating the attraction basin of the global minimum will also be very difficult, because of the sampler's difficulty in escaping from high-energy barriers.

The GWL algorithm works in a quite different style than the foregoing annealing-based algorithms. With an appropriate setting, it leads to a random walk in the energy space, and thus it

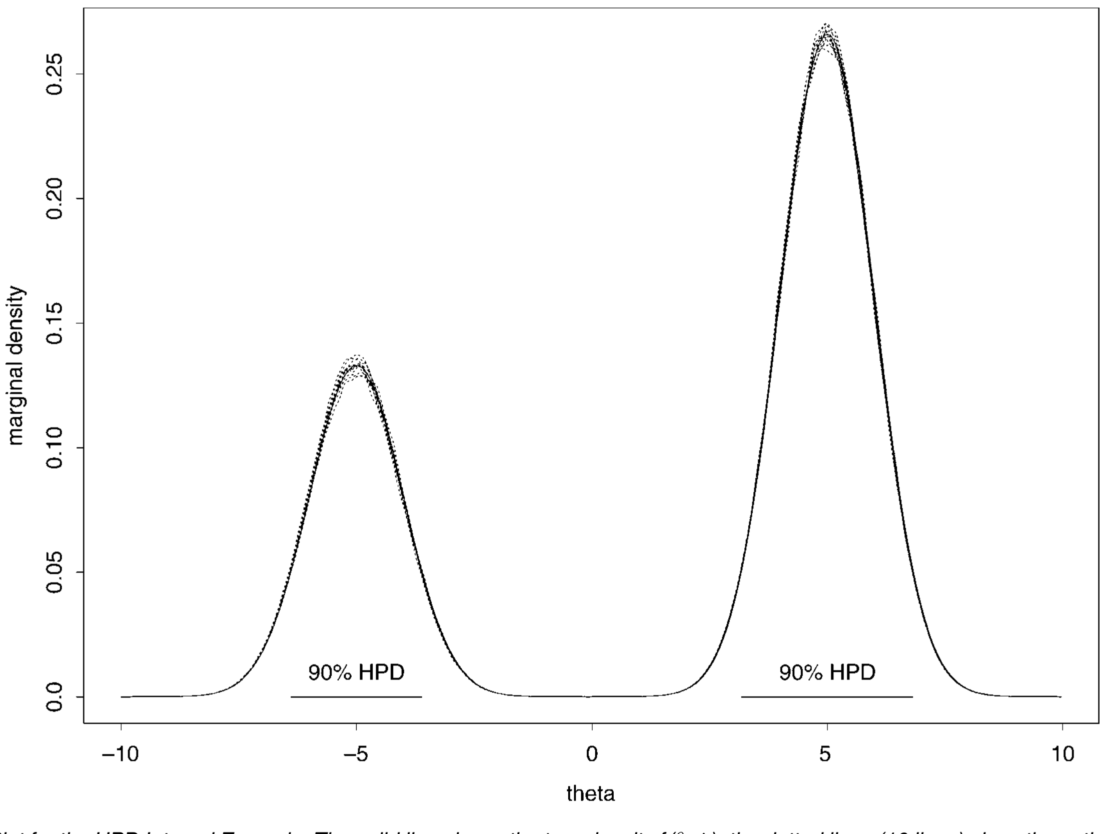

Figure 3. Plot for the HPD Interval Example. The solid line shows the true density f(θ, ϕ), the dotted lines (10 lines) show the estimates of f(θ, ϕ) obtained in the 10 runs, and the dashed line shows the estimate of f(θ, ϕ) by averaging over the 10 runs. The two segments at the bottom show the estimate of the 90% HPD interval.

can overcome any barriers of the energy landscape. The following example compares the behavior of the GWL and simulated annealing algorithms in minimizing a function with multiple minima.

6.1 A Multimodal Example

Consider minimizing the following function on $[-1.1, 1.1]^2$:

$$H(x,y) = -\big(x\sin(20y) + y\sin(20x)\big)^2 \cosh(\sin(10x)x) - \big(x\cos(10y) - y\sin(10x)\big)^2 \cosh(\cos(20y)y),$$

whose global minimum is -8.12465 attained at $(x,y) = (-1.0445, -1.0084)$ and $(1.0445, -1.0084)$. This example is modified from example 5.2.1 of Robert and Casella (1999). Figure 4 shows that $H(x,y)$ has a multitude of local minima separated by high-energy barriers. In what follows we form the problem so as to find the modes of the distribution

$$f(x,y) = \begin{cases} \dfrac{1}{Z}\exp\left(-\dfrac{H(x,y)}{t}\right) & \text{if } (x,y) \in [-1.1, 1.1]^2 \\ 0 & \text{otherwise,} \end{cases}$$

where $t = .1$ and Z is the unknown normalizing constant.

The GWL algorithm was first applied to this example. The sample space was partitioned into 41 subregions with equal energy bandwidths: $E_1 = \{(x,y) \in \mathcal{X} : H(x,y) \le -8.0\}$, $E_2 = \{(x,y) \in \mathcal{X} : -8.0 < H(x,y) \le -7.8\}, \ldots,$ and $E_{41} = \{(x,y) \in \mathcal{X} : -.2 < H(x,y) \le 0\}$. In simulations, we set $\psi(x,y) = \exp(-H(x,y)/.1)$, $\rho = 0$, and the proposal distribution as $N_2((x_k, y_k)', .3^2\mathbf{I}_2)$. The GWL algorithm was run for only one stage with 1,000 iterations. The overall acceptance rate was about .2, and the CPU time was about .03 second. Figure 5(a) shows the sample path of a typical run of the GWL algorithm. The central part of the sample space [Fig. 4(b)] has a big area, which is about half of the total area of the sample space, but it is seldom visited in the run. This is the fact due to that the GWL algorithm leads to a random walk in the energy space instead of the sample space. Hence the GWL algorithm can overcome any energy barrier of the energy landscape and locate the global energy minima very quickly. In fact, the GWL algorithm has located the two global minima many times within the 1,000 iterations.

For comparison, the MH algorithm was also applied to this example. Two runs were made with $t = 5$ and $t = .1$. The run with the high temperature mimics the simulation of simulated annealing and other tempering algorithms at high temperatures. The respective proposal distributions used in the two runs are $N_2((x_k, y_k)', .75^2\mathbf{I}_2)$ and $N_2((x_k, y_k)', .05\mathbf{I}_2)$. Each run consisted of 1,000 iterations and cost about .03 second of CPU time. The acceptance rates of the two runs were .38 and .25. Figures 5(b) and 5(c) show the sample paths of the two runs. Figure 5(b) shows that at the high temperature, the MH algorithm is almost equivalent to a random walk in the space $\mathcal{X}$. Because the sample space is usually very large, it is almost impossible for the algorithm to locate the global energy minimum at this stage. Figure 5(c) shows that at the low temperature, the MH algorithm tends to get trapped in one of local minima. It is also almost impossible for the algorithm to locate the global minimum at this stage.

Later, the GWL algorithm was rerun with $\delta_{\text{end}} = 10^{-4}$, $n_1 = 1{,}000$, $n_{s+1} = 1.1n_s$, and the proposal distribution $N((x_k, y_k)', .1^2\mathbf{I}_2)$. Here a smaller variance was chosen for the proposal distribution, because we would like to locate the global minimum precisely, instead of only the attraction basin of the global minimum. The simulation was repeated 100 times. The total CPU time was 3.6 seconds. The computational results are summarized in Table 3.

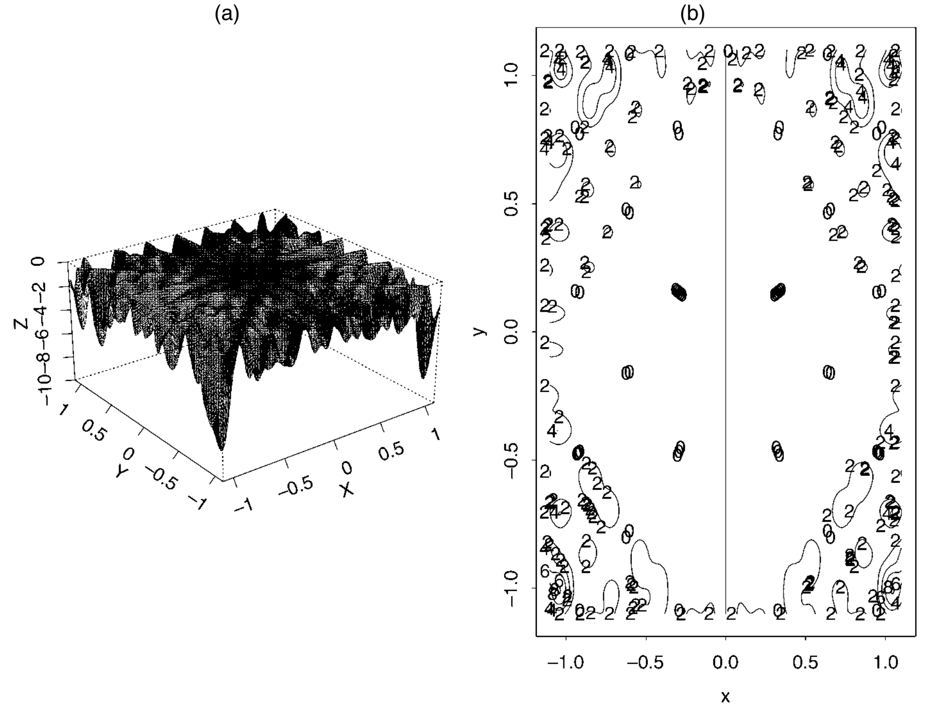

Figure 4. Contour (a) and Grid (b) Representations of the Function H(x, y) on [−1.1, 1.1]^2.

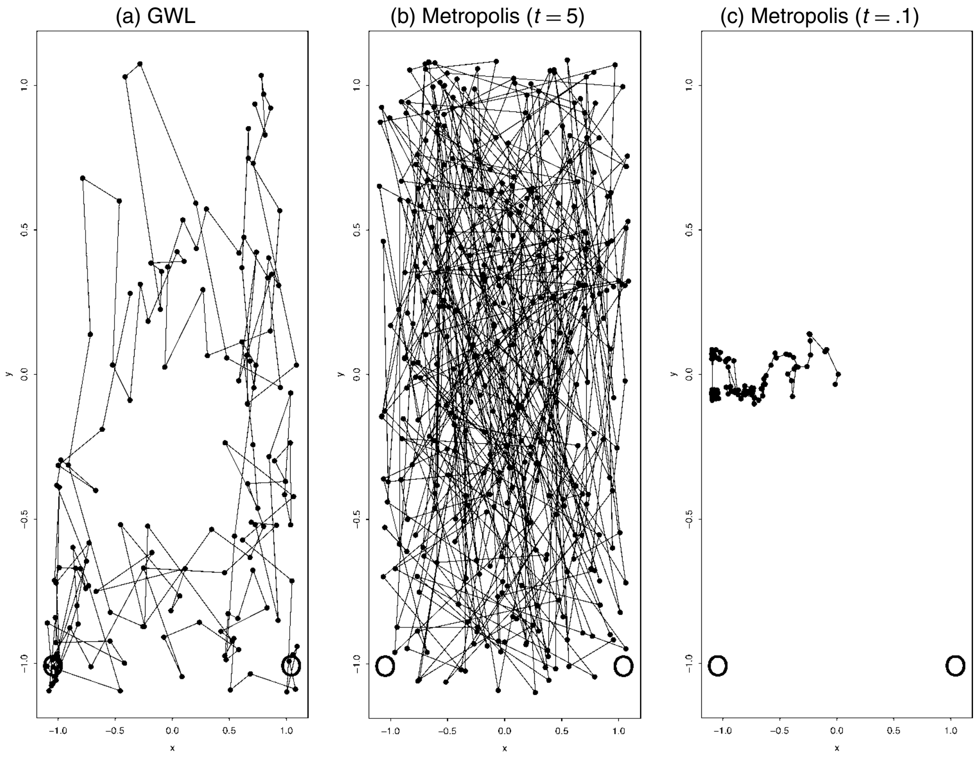

Figure 5. Sample Paths of the GWL and MH Algorithms. The circles in the plots show the locations of the two global minima. (a) The sample path of a GWL run. (b) The sample path of an MH run at t = 5. (c) The sample path of a MH run at t = .1.

For comparison, the simulated annealing algorithm was also applied to this example with various temperature schemes as given in Table 3. In all of the temperature schemes, the temperature decreased geometrically from the highest temperature t_{high} to the lowest temperature $t_{\mathrm{low}} = .1$ in a constant rate q_t; the number of iterations at the highest temperature was $n_{\mathrm{high}} = 120$, and it increased geometrically in a rate of $q_n = 1.05$ as the temperature decreased; the proposal distribution is $\mathrm{N}_2((x_k, y_k)', (.15t)^2\mathbf{I}_2)$. For each of the temperature schemes, the simulated annealing algorithm was run for 1,000 times independently. The overall acceptance rates of these runs varied between .22 and .32. The computational results are summarized in Table 3. The comparison shows that the GWL algorithm is superior to the simulated annealing algorithm for this simple example. To get the same proportion of the successful runs as that of the GWL algorithm, the simulated annealing algorithm needs about 12 folds of CPU time for this example.

6.2 Annealing GWL, ρ-GWL, and Neural Network Training

Because GWL (with $\rho = 0$) leads to a free random walk in the energy space, a natural question is whether efficiency of the GWL algorithm will be deteriorated due to oversampling from high-energy regions when the range of the energy function is wide. To avoid oversampling from high-energy regions, we make the following modification on the GWL algorithm. We call the modified algorithm "annealing GWL." In annealing

Table 3. Comparison of GWL and Simulated Annealing

Algorithm	*Mean*	*Standard error* ($\times 10^{-3}$)	*Minimum*	*Maximum*	*Proportion*	*Time (s)*
GWL	−8.12361	.03819	−8.12466	−8.11055	984	35.1
Annealing-1	−8.09763	2.70948	−8.12466	−6.64358	739	34.3
Annealing-2	−8.09579	2.80803	−8.12466	−7.14855	733	35.2
Annealing-3	−8.09018	3.68467	−8.12466	−6.26991	735	37.6
Annealing-4	−8.12216	.29299	−8.12466	−8.00916	879	125.6
Annealing-5	−8.12207	.36067	−8.12466	−7.93767	878	128.1
Annealing-6	−8.12109	.44673	−8.12466	−7.92996	872	130.7
Annealing-7	−8.11945	.86893	−8.12466	−7.70189	869	131.6
Annealing-8	−8.12443	.03289	−8.12466	−8.11187	986	446.5
Annealing-9	−8.12428	.05511	−8.12466	−8.10465	968	447.0
Annealing-10	−8.12433	.04209	−8.12466	−8.10643	982	449.9

NOTE: Let z_i, $i = 1, \ldots, 1{,}000$, denote the minimum energy value found in the ith run. "Mean" is the average of the z_i's, "standard error" is the standard deviation of "mean," "minimum" = $\min_{1 \le i \le 1{,}000} z_i$, "maximum" = $\max_{1 \le i \le 1{,}000} z_i$, and "proportion" = $\#\{i : z_i < -8.12\}$. In annealing-1, annealing-2, and annealing-3, the respective (t_{high}, q_t)'s are (50, .8831), (20, .8995), and (10, .9120), and the number of temperature levels is 51. In annealing-4, annealing-5, annealing-6, and annealing-7, the respective (t_{high}, q_t)'s are (50, .9205), (20, .9318), (10, .9404), and (5, .9492); and the number of temperature levels is 76. In annealing-8, annealing-9, and annealing-10, the respective (t_{high}, q_t)'s are (50, .9397), (20, .9484), and (10, .9550), and the number of temperature levels is 101.

GWL, we usually set $\psi(\mathbf{x}) = \exp(-H(\mathbf{x})/t)$ because it aims to minimize the function $H(\mathbf{x})$ instead of estimating an expectation. Note that for some problems (e.g., traveling salesman problems), we can set $\psi(\mathbf{x}) = H(\mathbf{x})$ provided that $\int |H(\mathbf{x})|\, d\mathbf{x} < \infty$ and $H(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathcal{X}$. Suppose that the sample space has been partitioned according to the energy function into m subregions, $E_1, \ldots, E_m$, and that the E_i's have been arranged in ascending order by energy; that is, if $i < j$, then $H(\mathbf{x}) < H(\mathbf{y})$ for any $\mathbf{x} \in E_i$ and $\mathbf{y} \in E_j$. Let $\varpi(z)$ denote the index of the subregion to which a sample $\mathbf{x}$ with energy $H(\mathbf{x}) = z$ belongs. For example, if $\mathbf{x} \in E_j$, then $\varpi(H(\mathbf{x})) = j$. Let $\mathcal{X}^{(s,k)}$ denote the sample space at iteration k of stage s of the simulation. The annealing GWL algorithm starts with $\mathcal{X}^{(1,1)} = \bigcup_{i=1}^m E_i$ and then iteratively sets

$$\mathcal{X}^{(s,k)} = \bigcup_{i=1}^{\varpi(H_{\min}+\Delta)} E_i, \tag{18}$$

where $H_{\min}$ is the minimum energy value obtained so far in the run and Δ is a user-specified parameter that controls the sample space of each iteration. The sample space $\mathcal{X}^{(s,k)}$ shrinks iteration by iteration. In this sense, we call the modified algorithm the annealing GWL algorithm. This algorithm can be run as the GWL algorithm except that its actual sample space needs to be updated every iteration or every number of iterations. It does not require that the simulation start from regions of specified energy. The performance of the annealing GWL algorithm depends on the value of Δ to some extent. If Δ is so large that $\mathcal{X}^{(s,k)} = \mathcal{X}$ for all iterations, then the annealing GWL algorithm turns to the GWL algorithm. In this case, it may take a long time to locate the global minimum. If Δ is too small, then $\mathcal{X}^{(s,k)}$ may contain too many disconnected regions in the later period of the simulation. In this case there will be too many rejections for the proposed moves, and it may also take a long time to locate the global minimum. We note that a similar idea was given by Lee and Cho (1994) in applying the multicanonical algorithm to traveling salesman problems.

The shrinkage of the sample space in the annealing GWL algorithm is equivalent to the following iterative modification on the working function $\psi(\mathbf{x})$. Set $\psi^{(1,0)} = \psi(\mathbf{x})$, and then update the working function iteratively as

$$\psi^{(s,k)}(\mathbf{x}) = \begin{cases} \psi(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{X}^{(s,k)} \\ 0 & \text{otherwise,} \end{cases}$$

where $\psi^{(s,k)}(\mathbf{x})$ is the working function at iteration k of stage s and $\mathcal{X}^{(s,k)}$ is as defined in (18). Hence if the proposal distribution is global, then Theorem 1 is still applicable to the annealing GWL algorithm; that is, as $s \to \infty$ and $k \to \infty$, we have $\hat{g}(E_i) \to cg(E_i)$ for $i = 1, \ldots, \varpi(H_{\min} + \Delta)$.

The ρ-GWL algorithm ($\rho > 0$) has also the ability to avoid oversampling from high-energy regions. It will bias sampling to low-energy regions. Potentially, the ρ-GWL algorithm is superior to the annealing GWL algorithm, because it always works on the entire sample space, whereas the annealing GWL algorithm may suffer from the difficulty in crossing disconnected regions in the latter stages of the simulation if the proposal function is not chosen appropriately. The difference between the ρ-GWL and annealing GWL algorithms can be summarized simply as follows: ρ-GWL puts a soft penalty on the visiting to high-energy regions, whereas annealing GWL uses a hard threshold penalty to prohibit visitation to high-energy regions. We compare these numerically through a neural network training example.

Consider a dataset of n observations, $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, where $\mathbf{x}_k = (x_{k1}, \ldots, x_{kp})$ is a p-dimensional vector and y_k is a real number. Suppose that $\mathbf{x}$ and y are nonlinearly related through an unknown function $r(\mathbf{x})$ as

$$y_k = r(\mathbf{x}_k) + \epsilon_k, \qquad k = 1, \ldots, n,$$

where the ϵ_k's are random errors. Multilayer perceptrons (MLPs) (Rumelhart and McClelland 1986) approximate the function $r(\mathbf{x})$ in a function of the form

$$\hat{r}(\mathbf{x}_k) = \varphi_2\left(\alpha_0 + \sum_{i=1}^{M} \alpha_i \varphi_1\left(\beta_{i0} + \sum_{j=1}^{p} \beta_{ij} x_{kj}\right)\right),$$

where the $\varphi_1(\cdot)$ and $\varphi(\cdot)$ are activation functions, M is the number of hidden units, and α's and β's are the connection weights, in the context of neural networks. In practice, $\varphi_1(\cdot)$, $\varphi_2(\cdot)$, and M are specified a priori, and the α's and β's can be determined by minimizing an objective function, such as

$$H(\alpha, \beta) = \sum_{k=1}^{n} (\hat{r}(\mathbf{x}_k) - y_k)^2. \tag{19}$$

So the problem of MLP training is an optimization problem.

MLP training has been a benchmark problem of optimization because of its high nonlinearity and high dimensionality. In this article we compare the performance of the annealing GWL, ρ-GWL, parallel tempering, and simulated annealing algorithm in training an MLP for a simplified two-spiral example (Lang and Witbrock 1988), where two spirals are intertwined and our task is to learn to determine which of the 120 training points (shown in Fig. 6) belong to which spiral. An MLP is specified for this example with $M = 20$ and $\varphi_1(z) = \varphi_2(z) = 1/(1 + e^{-z})$. This MLP has 81 connection weights to be determined by minimizing the energy function

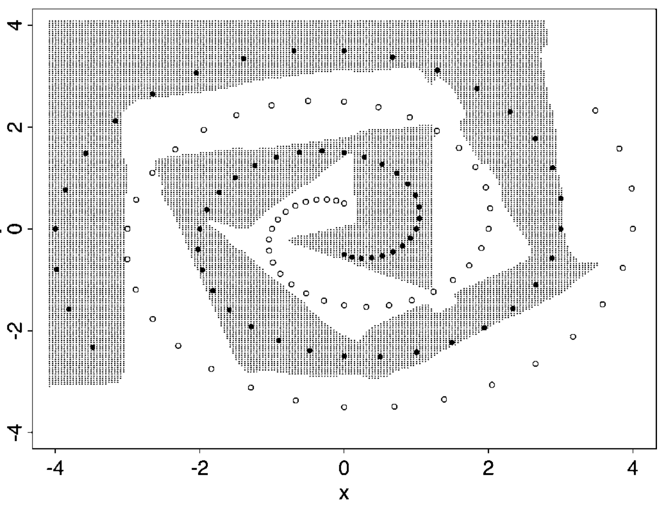

Figure 6. Learned Classification Boundary in One Run of the Annealing GWL Algorithm. The black and white points show the training data for two different spirals.

$H(\alpha, \beta)$ defined in (19); so the sample space for this example is $\mathcal{X} = \mathbb{R}^{81}$. Suppose that the sample space has been partitioned into 300 subregions with equal energy bandwidths: $E_1 = \{(\alpha, \beta) \in \mathbb{R}^{81} : H(\alpha, \beta) \le .2\}$, $E_2 = \{(\alpha, \beta) \in \mathbb{R}^{81} : .2 < H(\alpha, \beta) \le .4\}, \ldots,$ and $E_{300} = \{(\alpha, \beta) \in \mathbb{R}^{81} : H(\alpha, \beta) > 59.8\}$. The annealing GWL algorithm was first applied to this example. It was run with $\psi(\alpha, \beta) = \exp\{-H(\alpha, \beta)\}$, $n_1 = 10^5$, $n_{s+1} = 1.1 n_s$, and $\Delta = 5$. The proposal distribution used is a spherical distribution. A direction was first generated uniformly, and then the radius was drawn from N(0, 1). The simulation continues until a configuration with $H(\alpha, \beta) \le .1$ has been found or δ_s has been less than $\delta_{\text{end}} = .05$. The overall acceptance rate of the GWL moves is about .23. The algorithm was run 20 times independently. Figure 6 shows the learned classification boundary in one run. The MLP has separated the two spirals successfully. The numerical results are summarized in Table 4 together with the results produced by other algorithms, as we described later.

For each of the ρ values given in Table 4, the ρ-GWL algorithm was run 20 times independently with the same setting as that used by the annealing GWL algorithm. The annealing GWL and ρ-GWL (with $\rho > 1$) algorithms have almost the same performance. They all outperform the GWL and 1-GWL algorithms, which tend to oversample from high-energy regions. We note that the superiority of the 1-GWL algorithm to the GWL algorithm is consistent with the finding of Hesselbo and Stinchcombe (1995) that $1/k$-ensemble sampling is superior to the multicanonical algorithm in optimization.

For comparison, we also applied the parallel tempering and simulated annealing algorithms to this example. In parallel tempering, we set the number of temperature levels as $N = 25$, the highest temperature $t_{\text{high}} = 1.0, 5, 10$, the lowest temperature $t_{\text{low}} = .01$, and the other temperatures were equally spaced between t_{high} and t_{low} in the logarithm. Let $(\alpha^{(i)}, \beta^{(i)})$ denote a sample from the distribution

$$\pi_i(\alpha, \beta) \propto \exp\left(-\frac{H(\alpha, \beta)}{t_i}\right), \tag{20}$$

which is the trial distribution corresponding to the ith temperature level. At each iteration, the algorithm consists of two steps:

a. Update independently each $(\alpha^{(i)}, \beta^{(i)})$ for κ steps with the MH algorithm. A spherical proposal distribution was used in the MH moves. A direction was generated uniformly, and then the radius was drawn from $N(0, \sigma_i^2)$, where σ_i^2 was calibrated such that the updates had an appropriate acceptance rate. In simulation, we set $\kappa = 2$ and $\sigma_i^2 = \min\{5t_i, 5\}$.

b. Try to exchange $(\alpha^{(i)}, \beta^{(i)})$ with $(\alpha^{(j)}, \beta^{(j)})$ for N pairs with i being sampled at random from $\{1, 2, \ldots, N\}$ and $j = i \pm 1$ with probability $\omega(j|i)$, where $\omega(i+1|i) = \omega(i-1|i) = \frac{1}{2}$ and $\omega(2|1) = \omega(N-1|N) = 1$. The exchange was accepted with probability

$$\min\left\{1, \exp\left[\left(H(\alpha_i, \beta_i) - H(\alpha_j, \beta_j)\right)\left(\frac{1}{t_i} - \frac{1}{t_j}\right)\right]\right\},$$

according to the MH rule.

For each value of t_{high}, the algorithm was run 20 times independently. Each run consisted of 13,000 iterations. The overall acceptance rates of the MH moves were about .24, .4, and .45; the overall acceptance rates of the exchange moves were about .42, .41, and .4. These findings suggest that parallel tempering has been implemented effectively (Gelman, Roberts, and Gilks 1996). The other temperature schemes were also tried, including $t_{\text{high}} = 20$ and $t_{\text{low}} = .01$, and $t_{\text{high}} = .5$ and $t_{\text{low}} = .01$. The results were all similar or inferior to those reported in Table 4.

For the simulated annealing algorithm, we tried three temperature schemes. In scheme 1 we set $t_{\text{high}} = 1.0$, $t_{\text{low}} = .01$, the number of temperature levels $N = 50$, and the temperature decreasing factor $q = .91$. In scheme 2 we set $t_{\text{high}} = 5$, $t_{\text{low}} = .01$, $N = 60$, and $q = .9$. In scheme 3 we set $t_{\text{high}} = 10$, $t_{\text{low}} = .01$, $N = 70$, and $q = .905$. At each temperature level, the configuration was updated with the MH algorithm for n_i steps, where $n_1 = 3{,}000$ and $n_{i+1} = 1.05 n_i$. The proposal distribution was the same as that used in the simulated tempering algorithm. For each of the schemes, the algorithm was run 20 times independently. Table 4 shows that the annealing GWL and ρ-GWL algorithms significantly outperform the parallel tempering and simulated annealing algorithm. Form the "proportion" column, we can see that the ρ-GWL algorithm ($\rho > 1$) is able to find the global minimum in almost all of the 20 runs, whereas the parallel tempering algorithm is never able to find the global

Table 4. Comparison of the GWL, ρ-GWL, Annealing GWL, Parallel Tempering, and Simulated Annealing Algorithms for the MLP Training Example

Algorithm	*Mean*	*Standard error*	*Minimum*	*Maximum*	*Proportion*	*Time (s)*	*Stage*
GWL	9.53	1.08	0	19.68	1	347	5.0(5)
Annealing GWL	.05	.05	0	1.00	19	347	2.4(5)
1-GWL	1.90	.61	0	11.50	9	336	3.3(4)
1.5-GWL	.2	.16	0	3.01	18	336	2.3(4)
2-GWL	0	0	0	0	20	336	2.6(4)
3-GWL	.07	.05	0	1.01	18	336	2.5(4)
Tempering-1	3.99	.19	2.68	5.32	0	338	
Tempering-2	4.22	.16	2.61	5.50	0	338	
Tempering-3	4.55	.21	2.46	6.11	0	338	
Annealing-1	7.43	1.04	0	18.33	1	341	
Annealing-2	16.01	1.32	5.50	29.94	0	600	
Annealing-3	18.75	1.30	9.50	30.63	0	1,117	

NOTE: The 1-GWL, 1.5-GWL, 2-GWL, and 3-GWL correspond to the runs of ρ-GWL with $\rho = 1$, 1.5, 2, and 3. "Mean," "standard error," "minimum," and "maximum" are calculated as in Table 3 except based on 20 runs. "Proportion" denotes the number of runs (out of 20) during which one global minimum configuration with $H(\alpha, \beta) < .1$ was located. "Time" is the CPU time (in seconds) allowed for a single run. In the "stage" column, the numbers before and in the parentheses are the average of the number of stages used by the run and the maximum number of stages within the allowed CPU time.

minimum under the given settings, and the simulated annealing algorithm performs even worse than parallel tempering. For the parallel tempering and simulated annealing algorithms, we also tried other spherical proposals, including those with $\sigma_i^2 = \min\{2.5t_i, 2.5\}$ and $\sigma^2 = t_i$; the results were similar.

Table 4 also shows that the performance of the simulated annealing algorithm becomes worse as t_{high} increases and the running time increases. This is in contrast to our intuition of the higher the temperature and the longer the CPU time, the better the results. But our observation is generic to neural network training. This can be explained as follows. Let $(\hat\alpha, \hat\beta)$ denote a local energy minimum configuration located by the simulated annealing algorithm. As we know, the MH algorithm is almost equivalent to a random walk when the temperature is high. Due to the cumulative effect of the random walk, the magnitude of $(\hat\alpha, \hat\beta)$ located at high temperatures is often large. Note that no penalty is put on the magnitude of $(\hat\alpha, \hat\beta)$ in (20). For a configuration of large magnitude, if it is trapped into a local energy minimum, then it will be less likely to escape from the local minimum than will a configuration of small magnitude, because more iterations are needed for the former configuration to reduce the effect of the large-magnitude connection weights by changing their values. This observation is not new in neural computation; it has led to the approach of regularization (Bishop 1995), which penalizes the connection weights of large magnitude in neural network training.

Table 4 also shows that the parallel tempering algorithm is more robust than the simulated annealing algorithm to the variation of the temperature schemes and proposal distributions. In parallel tempering, there are multiple chains running in parallel. Once a local energy minimum is located at the high temperature, the corresponding configuration is exchanged to the low temperature level, preventing the magnitude of the configuration from growing any further.

7. DISCUSSION

In this article we have introduced the GWL algorithm and showed that it can be applied to many problems of Monte Carlo integration and optimization, including normalizing constant estimation, Bayesian model selection, HPD interval construction, function optimization, and others. The GWL algorithm has a number of features that conventional Monte Carlo algorithms do not have. First, it provides a new method for Monte Carlo integration based on stochastic approximation. A rigorous theory for the validity of the approximation is provided in this article. Second, it is an excellent tool for Monte Carlo optimization. With an appropriate setting, the GWL algorithm can lead to a random walk in the energy space. Hence the GWL algorithm has the ability sample relevant parts of the sample space, even in the presence of many local energy minima. Third, its self-adjusting nature makes the simulation converge very quickly. The convergence can be checked on-line in a single run, in contrast to the multiple-run checking required by conventional MCMC algorithms.

In what follows we discuss how to set the free parameters δ and ρ, the partition of the sample space, and the number of iterations for the ρ-GWL algorithm. Let λ_k denote the convergence rate of the ρ-GWL algorithm at iteration k of stage s. From the proof of Theorem 1 [see (A.5)], we know that the convergence rate of the iteration is

$$\lambda_{s,k} = \frac{1}{(1+\tau_0^{(s,k)})^2}, \qquad (21)$$

where $\tau_0^{(s,k)} = \frac{\delta_s}{S}\min_{1\le j\le m}\sum_{i=j}^m \rho^{i-j}\hat g^{(s,k)}(E_j)$ and $S = \sum_{j=1}^m g(E_j)$. The larger the value of $\lambda_{s,k}$, the slower the convergence. To simplify the discussion, we assume that the sample space is partitioned such that $\int_{E_1}\psi(\mathbf{x})\,d\mathbf{x} \approx \cdots \approx \int_{E_m}\psi(\mathbf{x})\,d\mathbf{x}$. Under this assumption,

$$\tau_0^{(s,k)} \approx \frac{\delta_s}{\sum_{j=1}^m\sum_{i=j}^m \rho^{i-j}} = \frac{(1-\rho)^2\delta_s}{\rho^{m+1}-(m+1)\rho+m}, \qquad (22)$$

where $0^0 = 1$ is defined to accommodate the case where $\rho = 0$. Figure 7 shows the curve of λ_k as a function of ρ and m by substituting (22) into (21).

First, we note that $\lambda_{s,k}$ is an increasing function of ρ. The smaller the ρ, the faster the convergence. This is consistent with our results in Section 4, which show numerically that the 1-GWL algorithm has a slower convergence than the GWL algorithm. This suggests that we should set $\rho = 0$ if we aim to estimate the $g(E_i)$'s. But if we aim for optimization, then we may set $\rho > 0$ to bias the simulation to low-energy regions.

Second, we note that $\lambda_{s,k}$ is an increasing function of m. The smaller the m, the faster the convergence. This conclusion is drawn under the assumption that the entire sample space can be well explored with a partition of m subregions. In practice, however, this may not be the case. If m is too small, then the sample space may not be able to be well explored. We note again that the GWL algorithm is reduced to the MH algorithm within the same subregion. Hence, in partitioning the sample space, a trade-off should be made between the number of subregions and the feasibility of sampling within subregions. For example, if the sample space is partitioned according to the energy function, then it is reasonable to partition the sample space such that $\sup_{E_i}H(\mathbf{x}) - \inf_{E_i}H(\mathbf{x}) \approx \tau$ for $1 \le i \le m$, where $\sup_{E_i}H(\mathbf{x}) - \inf_{E_i}H(\mathbf{x})$ is the range of energy on the subregion E_i. If $\psi(\mathbf{x}) = \exp\{-H(\mathbf{x})/\tau\}$ is set in the run, then the

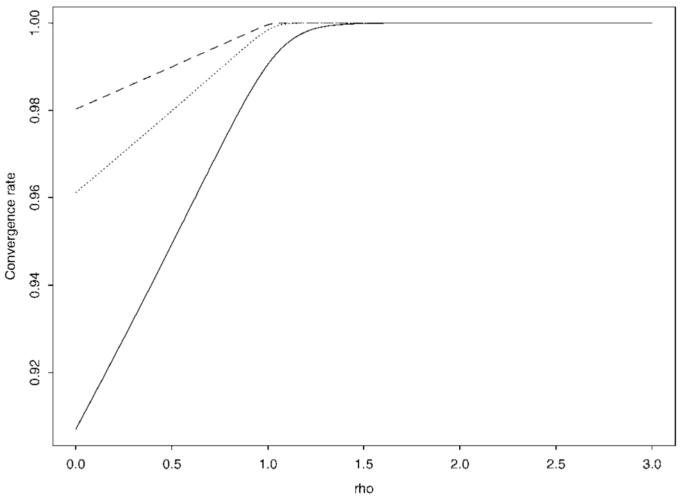

Figure 7. Convergence Rate of ρ-GWL Plotted as a Function of ρ and m (——, $m = 20$; $\cdots\cdot$, $m = 50$; ---, $m = 100$). In the plot δ_s is fixed to 1.

MH moves within the same subregion will have a reasonable acceptance rate, ranging approximately from $1/e$ to 1.

Third, we note that λ_k is a decreasing function of δ_s. The larger the δ_s, the faster the convergence. It is shown in Corollary 1 that at each stage the mean squared error of the estimates is on the order of $O(\delta_s)$ as the number of iterations tends to infinity. These observations suggest that δ should be set to a large value in the early stages of the simulation, and it should decrease as the simulation moves toward convergence. A large value of δ in the early stages will force all subregions to be visited very quickly.

Fourth, the number of iterations at each stage can be roughly determined based on (A.7) in the Appendix, where the term (I) measures the reminder effect of the initial estimates. For simplicity, we assume that $a_l \equiv a$ for each l, where a is a constant close to 1. For a given tolerant error ϵ, the number of iterations at stage s should be chosen such that $(1-a)^{K_s} < \epsilon$ or, equivalently, $K_s > -\log(\epsilon)/\delta_s$. Hence, roughly, we should have

$$\frac{K_{s+1}}{K_s} > \frac{\delta_s}{\delta_{s+1}}.$$

For example, if δ_s decreases geometrically with rate γ, then K_s should increase geometrically with rate $1/\gamma$.

APPENDIX: PROOFS

Proof of Theorem 1

For simplicity, we denote $g(E_i)$ by g_i and denote $\hat{g}^{(s,k)}(E_i)$ by $\hat{g}_i^{(k)}$ in the proof. Here the superscript s is omitted, because the calculation is for only one stage. Without loss of generality, we assume that $S = \sum_{i=1}^m g_i$ is known, $\sum_{i=1}^m \hat{g}_i^{(k)} = S$, and the sample of the next iteration $\mathbf{x}_{k+1} \in E_j$. To make $\sum_{i=1}^m \hat{g}_i^{(k+1)} = S$, we set

$$\hat{g}_i^{(k+1)} = \begin{cases} \dfrac{\hat{g}_i^{(k)}}{1+\tau_j^{(k)}} & \text{for } i = 1,\ldots,j-1 \\[2ex] \dfrac{\hat{g}_i^{(k)} + \delta_s \rho^{i-j}\hat{g}_j^{(k)}}{1+\tau_j^{(k)}} & \text{for } i = j,\ldots,m, \end{cases} \tag{A.1}$$

where $\tau_j^{(k)} = \epsilon_j \hat{g}_j^{(k)}/S > 0$ and $\epsilon_j = \delta_s \sum_{i=j}^m \rho^{i-j}$. The weight adjustment by the multiplication factor $(1+\tau_j^{(k)})$ will not change the process of the simulation. Given $\hat{g}_i^{(k)}$, $i = 1,\ldots,m$, the acceptance of the GWL move is guided by (7), so that

$$P(\mathbf{x}_{k+1} \in E_j) = \frac{1}{A_k}\frac{\int_{E_j}\psi(\mathbf{x})\,d\mathbf{x}}{\hat{g}_j^{(k)}}, \qquad j = 1,\ldots,m,$$

where $A_k = \sum_{j=1}^m \int_{E_j}\psi(\mathbf{x})\,d\mathbf{x}/\hat{g}_j^{(k)}$ is the normalizing constant of this distribution.

Let $\tau_0^{(k)} = \min_{1\le j\le m}\tau_j^{(k)}$, and $D_k = \sum_{i=1}^m (\hat{g}_i^{(k)} - g_i)^2$. Then we have the following calculation:

$$\begin{aligned}
&E\big(D_{k+1}\big|\hat{g}_i^{(k)}, i = 1,\ldots,m\big) \\
&\quad = \sum_{j=1}^m \Bigg\{\sum_{i=1}^{j-1}\bigg(\frac{\hat{g}_i^{(k)}}{1+\tau_j^{(k)}} - g_i\bigg)^2 \\
&\qquad + \sum_{i=j}^m \bigg(\frac{\hat{g}_i^{(k)} + \delta_s\rho^{i-j}\hat{g}_j^{(k)}}{1+\tau_j^{(k)}} - g_i\bigg)^2\Bigg\} P(\mathbf{x}_{k+1} \in E_j) \\
&\quad \le \frac{1}{(1+\tau_0^{(k)})^2} \\
&\qquad \times \sum_{j=1}^m \Bigg\{\sum_{i=1}^{j-1}\big(\hat{g}_i^{(k)} - g_i\big)^2 \\
&\qquad\qquad + \sum_{i=1}^{j-1}\big(\tau_j^{(k)}\big)^2 g_i^2 - 2\sum_{i=1}^{j-1}\tau_j^{(k)} g_i\big(\hat{g}_i^{(k)} - g_i\big) \\
&\qquad\qquad + \sum_{i=j}^m \big(\hat{g}_i^{(k)} - g_i\big)^2 + \sum_{i=j}^m \big(\delta_s\rho^{i-j}\hat{g}_j^{(k)} - \tau_j^{(k)} g_i\big)^2 \\
&\qquad\qquad + 2\sum_{i=j}^m \big(\hat{g}_i^{(k)} - g_i\big)\big(\delta_s\rho^{i-j}\hat{g}_j^{(k)} - \tau_j^{(k)} g_i\big)\Bigg\} \\
&\qquad \times P(\mathbf{x}_{k+1} \in E_j) \\
&\quad \le \frac{D_k}{(1+\tau_0^{(k)})^2} + \sum_{i=1}^m g_i^2 \sum_{j=1}^m \big(\tau_j^{(k)}\big)^2 P(\mathbf{x}_{k+1} \in E_j) \\
&\qquad - 2\sum_{i=1}^m g_i\big(\hat{g}_i^{(k)} - g_i\big)\sum_{j=1}^m \tau_j^{(k)} P(\mathbf{x}_{k+1} \in E_j) \\
&\qquad + \delta_s^2 \sum_{j=1}^m\sum_{i=j}^m \big(\rho^{i-j}\hat{g}_j^{(k)}\big)^2 P(\mathbf{x}_{k+1} \in E_j) \\
&\qquad + 2\delta_s \sum_{j=1}^m\sum_{i=j}^m \big(\hat{g}_i^{(k)} - g_i\big)\rho^{i-j}\hat{g}_j^{(k)} P(\mathbf{x}_{k+1} \in E_j).
\end{aligned} \tag{A.2}$$

Because of the relations

$$\begin{aligned}
&\sum_{j=1}^m \tau_j^{(k)} P(\mathbf{x}_{k+1} \in E_j) \\
&\quad = \frac{1}{A_k S}\sum_{j=1}^m \epsilon_j \int_{E_j}\psi(\mathbf{x})\,d\mathbf{x} = \frac{\delta_s}{A_k S}\sum_{j=1}^m\sum_{i=j}^m \rho^{i-j}\int_{E_j}\psi(\mathbf{x})\,d\mathbf{x} \\
&\quad = \frac{\delta_s}{A_k S}\sum_{i=1}^m\sum_{j=1}^i \rho^{i-j}\int_{E_j}\psi(\mathbf{x}) = \frac{\delta_s}{A_k S}\sum_{i=1}^m g_i = \frac{\delta_s}{A_k}
\end{aligned} \tag{A.3}$$

and

$$\begin{aligned}
&\sum_{j=1}^m\sum_{i=j}^m \big(\hat{g}_i^{(k)} - g_i\big)\rho^{i-j}\hat{g}_j^{(k)} P(\mathbf{x}_{k+1} \in E_j) \\
&\quad = \sum_{i=1}^m\sum_{j=1}^i \big(\hat{g}_i^{(k)} - g_i\big)\rho^{i-j}\frac{\int_{E_j}\psi(\mathbf{x})\,d\mathbf{x}}{A_k} \\
&\quad = \frac{1}{A_k}\sum_{i=1}^m g_i\big(\hat{g}_i^{(k)} - g_i\big),
\end{aligned} \tag{A.4}$$

the third and fifth terms of the last inequality of (A.2) cancel each other. Because $\tau_j^{(k)}$ has the same order as δ_s, that is, $\tau_j^{(k)} \sim O(\delta_s)$, we have

$$E\big(D_{k+1}\big|\hat{g}_i^{(k)}, i = 1,\ldots,m\big) \le \frac{1}{(1+\tau_0^{(k)})^2}D_k + O(\delta_s^2),$$

which implies that

$$E(D_{k+1}) \le \frac{1}{(1+\tau_0^{(k)})^2}E(D_k) + O(\delta_s^2) \tag{A.5}$$

and

$$E(D_k) \to 0 \quad \text{as } k \to \infty \text{ and } \delta_s \to 0.$$

Because $D_k > 0$, we know that

$$D_k \to 0 \quad \text{in probability,} \tag{A.6}$$

and then

$$\hat{g}^{(s,k)}(E_i) \to g(E_i) \quad \text{in probability,}$$

as $\delta_s \to 0$ and $k \to \infty$.

Proof of Corollary 1. Here we use the same notations as in the proof of Theorem 1. Because $\tau_0^{(k)} \sim O(\delta_s)$, that is, $\tau_0^{(k)}$ and δ_s are in the same order, we can write

$$\frac{1}{(1+\tau_0^{(k)})^2} \approx 1 - a_k\delta_s$$

in a Taylor expansion, where $a_k > 0$ and $a_k \sim O(1)$. From the recursive relationship given in (A.5), we have

$$\begin{aligned} ED_{k+1} &\le (1-a_k\delta_s)ED_k + O(\delta_s^2) \\ &\le (1-a_k\delta_s)[(1-a_{k-1}\delta_s)ED_{k-1} + O(\delta_s^2)] + O(\delta_s^2) \\ &\le \cdots \\ &\le \prod_{l=1}^{k}(1-a_l\delta_s)ED_1 + \left[1 + \sum_{l=2}^{k}\prod_{j=l}^{k}(1-a_j\delta_s)\right]O(\delta_s^2) \\ &= \text{(I)} + \text{(II)}. \end{aligned} \tag{A.7}$$

For part (I), it is easy to see that

$$\prod_{l=1}^{k}(1-a_l\delta_s) \to 0, \tag{A.8}$$

as $k \to \infty$.

For part (II), let

$$\begin{aligned} C &= 1 + \sum_{l=2}^{k}\prod_{j=l}^{k}(1-a_j\delta_s) \\ &= 1 + (1-a_k\delta_s)\left[1 + \sum_{l=2}^{k-1}\prod_{j=l}^{k-1}(1-a_j\delta_s)\right]. \end{aligned} \tag{A.9}$$

As $k \to \infty$, we have

$$C \le 1 + (1-a_k\delta_s)C, \tag{A.10}$$

which implies that $C = O(\frac{1}{\delta_s})$.

Summarizing the foregoing analysis for (I) and (II), we have

$$ED_k \le 0 + O\left(\frac{1}{\delta_s}\right)O(\delta_s^2) = O(\delta_s),$$

as $k \to \infty$.

It follows from (A.8) that the convergence of ED_k is geometric and the convergence rate varies slightly from iteration to iteration. The proof of the corollary is completed.

[Received October 2003. Revised February 2005.]

REFERENCES

Berg, B. A., and Neuhaus, T. (1991), "Multicanonical Algorithms for 1st Order Phase-Transitions," *Physics Letters B*, 267, 249–253.

Bishop, C. M. (1995), *Neural Networks for Pattern Recognition*, Oxford, U.K.: Oxford University Press.

Chen, M. H., and Shao, Q. M. (1999), "Monte Carlo Estimation of Bayesian Credible and HPD Intervals," *Journal of Computational and Graphical Statistics*, 8, 69–92.

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

Gelman, A., Roberts, R. O., and Gilks, W. R. (1996), "Efficient Metropolis Jumping Rules," in *Bayesian Statistics 5*, eds. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, New York: Oxford University Press, pp. 599–608.

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

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

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

Hansmann, U. H. E., and Okamoto, Y. (1997), "Numerical Comparisons of Three Recently Proposed Algorithms in the Protein Folding Problems," *Journal of Computational Chemistry*, 18, 920–933.

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

Hesselbo, B., and Stinchcombe, R. B. (1995), "Monte Carlo Simulation and Global Optimization Without Parameters," *Physical Review Letters*, 74, 2151–2155.

Hyndman, R. J. (1996), "Computing and Graphing Highest Density Regions," *The American Statistician*, 50, 120–126.

Kass, R. E., and Wasserman, L. (1996), "The Selection of Prior Distributions by Formal Rules," *Journal of the American Statistical Association*, 91, 1343–1370.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983), "Optimization by Simulated Annealing," *Science*, 220, 671–680.

Lang, K. J., and Witbrock, M. J. (1988), "Learning to Tell Two Spirals Apart," in *Proceedings of the 1988 Connectionist Models*, eds. D. Touretzky, G. Hinton, and T. Sejnowski, San Mateo, CA: Morgan Kaufmann, pp. 52–59.

Lee, J., and Cho, M. Y. (1994), "Optimization by Multicanonical Annealing and the Traveling Salesman Problem," *Physical Review E*, 50, 651–654.

Liang, F. (2002), "Dynamically Weighted Importance Sampling in Monte Carlo Computation," *Journal of the American Statistical Association*, 97, 807–821.

Liang, F., and Wong, W. H. (2001), "Real Parameter Evolutionary Monte Carlo With Applications in Bayesian Mixture Models," *Journal of the American Statistical Association*, 96, 653–666.

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

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

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

Rathore, N., and de Pablo, J. J. (2002), "Monte Carlo Simulation of Proteins Through a Random Walk in Energy Space," *Journal of Chemical Physics*, 116, 7225–7230.

Robert, C. P., and Casella, G. (1999), *Monte Carlo Statistical Methods*, New York: Springer-Verlag.

Roberts, G. O. (1996), "Markov Chain Concepts Related to Sampling Algorithms," in *Markov Chain Monte Carlo in Practice*, eds. W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, London: Chapman & Hall, pp. 45–58.

Rumelhart, D., and McClelland, J. (1986), *Parallel Distributed Processing*, Cambridge, MA: MIT Press.

Swendsen, R. H., and Wang, J. S. (1987), "Nonuniversal Critical Dynamics in Monte Carlo Simulations," *Physical Review Letters*, 58, 86–88.

Tanner, M. A. (1996), *Tools for Statistical Inference* (3rd ed.), New York: Springer-Verlag.

Tierney, L. (1994), "Markov Chains for Exploring Posterior Distributions" (with discussion), *The Annals of Statistics*, 22, 1701–1786.

Wang, F., and Landau, D. P. (2001), "Efficient, Multiple-Range Random Walk Algorithm to Calculate the Density of States," *Physical Review Letters*, 86, 2050–2053.

Wei, G. C. G., and Tanner, M. A. (1990), "Calculating the Content and Boundary of the Highest Posterior Density Region via Data Augmentation," *Biometrika*, 77, 649–652.

Yamaguchi, C., and Okabe, Y. (2001), "Three-Dimensional Antiferromagnetic *q*-State Potts Models: Application of the Wang–Landau Algorithm," *Journal of Physics A*, 34, 8781–8794.