Annealing contour Monte Carlo algorithm for structure optimization in an off-lattice protein model

Faming Liang[a)]
Department of Statistics, Texas A&M University, College Station, Texas 77843-3143

(Received 4 December 2003; accepted 12 January 2004)

We present a space annealing version for a contour Monte Carlo algorithm and show that it can be applied successfully to finding the ground states for an off-lattice protein model. The comparison shows that the algorithm has made a significant improvement over the pruned-enriched-Rosenbluth method and the Metropolis Monte Carlo method in finding the ground states for AB models. For all sequences, the algorithm has renewed the putative ground energy values in the two-dimensional AB model and set the putative ground energy values in the three-dimensional AB model. © *2004 American Institute of Physics.* [DOI: 10.1063/1.1665529]

I. INTRODUCTION

Predicting the native structure of a protein from its sequence is one of the most challenging problems in biophysics. The difficulty of the problem comes from two aspects: First, the dimension of the system is usually high. It is in the same order with the number of atoms involved in the system. Hence, the phase space of the system is usually huge. Second, the energy landscape of the system is complex. The energy landscape can be characterized by a multitude of local minima separated by high-energy barriers. At low temperatures, conventional Monte Carlo and molecular dynamic simulations tend to get trapped in local minima, rendering the simulation ineffective. Given the complexity of the problem, there has been an increasing interest in understanding the relevant mechanics of the protein folding process by studying simplified models in recent years. For example, the HP model[1] treats each amino acid as a point particle and restricts the model to fold on a regular (quadratic or cubic) lattice. Even for this highly simplified model it is far from trivial to predict the native structure for a given sequence.[2–8]

We note that the algorithms proposed for the HP model may not be able to extend to an off-lattice protein model. Even if some of them are able to do so, they may not be efficient. In fact, in today's literature there are very few simple off-lattice models with known lowest-energy states that can be used as benchmarks for efficient algorithms, as noted by Hsu *et al.*[9] In this paper we propose the annealing contour Monte Carlo (ACMC) algorithm for an off-lattice model, so called the AB model.[10,11] The contour Monte Carlo (CMC) algorithm[12] is a generalization of the Wang–Landau algorithm[13] and the 1/*k*-ensemble algorithm.[14] It can be used for both continuous and discrete systems. The self-adjusting nature of the algorithm makes it be able to overcome any barrier of the energy landscape, so it is an excellent tool for Monte Carlo optimization. The ACMC algorithm is an accelerated version of the CMC algorithm for optimization problems. Our numerical results show that the ACMC is algorithm very promising for finding the ground states of proteins.

II. MODELS

The AB model consists of only two types of monomers, A and B, which behave as hydrophobic ($\sigma_i=+1$) and hydrophilic ($\sigma_i=-1$) monomers, respectively. The monomers are linked by rigid bonds of unit length to form linear chains residing in two- or three-dimensional space.

A. Two-dimensional AB model

In the two-dimensional (2D) AB model, the shape of *N*-mer is either specified by the $N-1$ bond vectors u_i or by $N-2$ bond angles $\theta_2,\ldots,\theta_{N-1}$. The energy function[10,11] consists of two types of contributions: bond angle and Lennard-Jones. It can be written as

$$H(\boldsymbol{u},\sigma)=\sum_{i=1}^{N-2} V_\theta(i)+\sum_{i=1}^{N-2}\sum_{j=i+2}^{N} V_{LJ}(r_{ij},\sigma_i,\sigma_j), \qquad (1)$$

where

$$V_\theta(i)=\tfrac{1}{4}(1-\boldsymbol{u}_i\cdot\boldsymbol{u}_{i+1}),$$

$$V_{LJ}(r_{ij},\sigma_i,\sigma_j)=4[r_{ij}^{-12}-C_2(\sigma_i,\sigma_j)r_{ij}^{-6}].$$

Here u_i is the unit-length bond vector joining monomer *i* to monomer $i+1$, r_{ij} denotes the distance between monomers *i* and *j*, and $C_2(\sigma_i,\sigma_j)$ is $+1$, $+\frac{1}{2}$, and $-\frac{1}{2}$, respectively, for AA, BB, and AB pairs, giving strong attraction between AA pairs, weak attraction between BB pairs, and weak repulsion between A and B.

B. Three-dimensional AB model

In the three-dimensional (3D) AB model, the shape of *N*-mer is either specified by the $N-1$ bond vectors $\boldsymbol{u}_i$ or by $N-2$ bond angles $\theta_2,\ldots,\theta_{N-1}$ and $N-3$ torsional angles $\tau_3,\ldots,\tau_{N-1}$. The energy function[15] consists of three types of contributions: bond angle, torsional angle, and Lennard-Jones. It can be written as

[a)]Electronic mail: fliang@stat.tamu.edu

$$H(\boldsymbol{u},\sigma)=\sum_{i=1}^{N-2} V_\theta(i)+\sum_{i=1}^{N-3} V_\tau(i)+\sum_{i=1}^{N-2}\sum_{j=i+2}^{N} V_{LJ}(r_{ij},\sigma_i,\sigma_j), \quad (2)$$

where

$$V_\theta(i)=\boldsymbol{u}_i\cdot\boldsymbol{u}_{i+1},$$

$$V_\tau(i)=-\tfrac{1}{2}\boldsymbol{u}_i\cdot\boldsymbol{u}_{i+2},$$

$$V_{LJ}(r_{ij},\sigma_i,\sigma_j)=4C_3(\sigma_i,\sigma_j)(r_{ij}^{-12}-r_{ij}^{-6}).$$

Here r_{ij} denotes the distance between monomers i and j, and $C_3(\sigma_i,\sigma_j)$ is $+1$ for AA pairs and $+\frac{1}{2}$ for BB and AB pairs. So in the 3D AB model, all nonbonded interactions are attractive, but with AA interactions carrying the highest weight. Since all nonadjacent monomer pairs attract each other, the model's tendency to form compact globular structures is further enhanced by this energy function.

The AB model has been studied in several papers.[7,10,11,15–18] The methods used to find the low-energy states include neural networks,[10] Metropolis Monte Carlo simulations,[11] simulated tempering,[17] multicanonial method,[15,16] biologically motivated methods,[18] and chain growth algorithm.[7] For the 2D case, the putative ground states are given in Refs. 7, 10, 11, and 18 for various AB sequences and for various sequence lengths, while for the 3D case, the putative ground states were not given at all or for very short sequences only.

III. ANNEALING CONTOUR MONTE CARLO ALGORITHM

In this section we first describe the general CMC algorithm and then present its space annealing version for optimization problems. Suppose that we want to make an inference for the Boltzmann distribution

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

where τ is called the temperature of the system, $\mathcal{X}$ is called the phase space, and $Z=\int_{\mathcal{X}}\exp\{-H(\boldsymbol{x})/\tau\}d\boldsymbol{x}$ is the partition function. The CMC algorithm works in the style of importance sampling; that is, it may work directly on any nonnegative function $\psi(\boldsymbol{x})$ defined on the phase space $\mathcal{X}$, instead on $f(\boldsymbol{x})$ necessarily, although we often set $\psi(\boldsymbol{x})=\exp\{-H(\boldsymbol{x})/\tau\}$ for convenience. For the function $\psi(\boldsymbol{x})$, we usually require that $\int_{\mathcal{X}}\psi(\boldsymbol{x})d\boldsymbol{x}<\infty$. Let $T(\cdot\to\cdot)$ denote a proposal distribution which is not necessarily symmetric. Suppose that the phase space has been partitioned into m mutually nonoverlapped subregions according to some criterion chosen by the user. For example, in all simulations of this paper we partition the phase space according to the energy function and index the subregions by $E_1,\ldots,E_m$ in ascending order of energy; that is, for any $\boldsymbol{x}\in E_i$ and $\boldsymbol{y}\in E_j$, if $i<j$, then $H(\boldsymbol{x})<H(\boldsymbol{y})$. Here we emphasize that these E_i's are nonoverlapping divisions of the phase space. Let $g(E_i)$ denote the weight associated with the subregion E_i. The $g(E_i)$ is determined by our parameter setting and phase-space partition as described below. The CMC simulation proceeds in several stages.

Let $\boldsymbol{x}_{s,k}$ and $\hat{g}^{(s,k)}(E_i)$ denote the sample and the estimate of $g(E_i)$, respectively, at the kth iteration of the sth stage of the simulation. In the first stage ($s=1$), the simulation starts with the initial estimates $\hat{g}^{(0,0)}(E_1)=\cdots=\hat{g}^{(0,0)}(E_m)=1$ and a random sample $\boldsymbol{x}_{0,0}(k=0)$, and then iterates as follows.

(a) Propose a new configuration $\boldsymbol{x}^*$ in the neighborhood of $\boldsymbol{x}_{s,k}$ according to a prespecified distribution $T(\cdot\to\cdot)$.

(b) Accept $\boldsymbol{x}^*$ with probability

$$\min\left\{\frac{\hat{g}^{(s,k)}(E_{I_{\boldsymbol{x}_{s,k}}})}{\hat{g}^{(s,k)}(E_{I_{\boldsymbol{x}^*}})}\frac{\psi(\boldsymbol{x}^*)}{\psi(\boldsymbol{x}_{s,k})}\frac{T(\boldsymbol{x}^*\to\boldsymbol{x}_{s,k})}{T(\boldsymbol{x}_{s,k}\to\boldsymbol{x}^*)},1\right\}, \quad (3)$$

where $I_{\boldsymbol{z}}$ denotes the index of the subregion where $\boldsymbol{z}$ belongs to. If it is accepted, set $\boldsymbol{x}_{s,k+1}=\boldsymbol{x}^*$ and $\hat{g}^{(s,k+1)}(E_{I_{\boldsymbol{x}_{s,k+1}}+i})=\hat{g}^{(s,k)}(E_{I_{\boldsymbol{x}_{s,k+1}}+i})+\delta_s\rho^i\hat{g}^{(s,k)}(E_{I_{\boldsymbol{x}_{s,k+1}}})$ for $i=0,\ldots,m-I_{\boldsymbol{x}_{s,k+1}}$; otherwise, set $\boldsymbol{x}_{s,k+1}=\boldsymbol{x}_{s,k}$ and $\hat{g}^{(s,k+1)}(E_{I_{\boldsymbol{x}_{s,k}}+i})=\hat{g}^{(s,k)}(E_{I_{\boldsymbol{x}_{s,k}}+i})+\delta_s\rho^i\hat{g}^{(s,k)}(E_{I_{\boldsymbol{x}_{s,k}}})$ for $i=0,\ldots,m-I_{\boldsymbol{x}_{s,k}}$.

The algorithm iterates until a stable histogram has been produced in the space of subregions. A histogram is said to be stable if its shape will not change much with more samples (a statistic is defined below to measure the stability of the histogram). Once the histogram is stable, we will reset the histogram, reduce the modification factor δ_s to a smaller value, set $s\leftarrow s+1$, and proceed to the next stage simulation. In the new stage simulation, we set $\boldsymbol{x}_{s,0}=\boldsymbol{x}_{s-1,K_{s-1}}$ and $\hat{g}^{(s,0)}(E_i)=\hat{g}^{(s-1,K_{s-1})}(E_i)$ for $i=1,\ldots,m$, where K_{s-1} is the total number of iterations performed in stage $s-1$. Note that the setting $\boldsymbol{x}_{s,0}=\boldsymbol{x}_{s-1,K_{s-1}}$ will not affect the simulation much. Theoretically, we can set $\boldsymbol{x}_{s,0}$ to a random sample as in the first stage. The parameter $\rho\geqslant 0$ is a user set parameter. The modification factor δ_1 is usually set to a large number—for example, 1 or 2—which allows us to reach all subregions very quickly even for a large system. In the followed stages, it will be reduced monotone in a function like $\delta_{s+1}=\gamma\delta_s$ with $\gamma<1$, or $\delta_{s+1}=\sqrt{1+\delta_s}-1$. The algorithm will run until δ_s has been reduced to a very small value—for example, less than 10^{-8}. In this paper, we set $\delta_1=e-1$ and $\delta_{s+1}=\sqrt{1+\delta_s}-1$ in all simulations.

About the convergence of the algorithm, we have the following theorem, of which proof is presented in the Appendix as a supporting document.

As $\delta_s\to 0$ and $k\to\infty$, for any non-negative function $\psi(\boldsymbol{x})$ with $\int_{\mathcal{X}}\psi(\boldsymbol{x})d\boldsymbol{x}<\infty$, we have

$$\hat{g}^{(s,k)}(E_i)\to g(E_i)=c\sum_{j=1}^{i}\rho^{i-j}\int_{E_j}\psi(\mathbf{x})d\boldsymbol{x}=c\left[\int_{E_i}\psi(x)d\boldsymbol{x}+\rho g(E_{i-1})\right], \quad (4)$$

in probability for $i=1,\ldots,m$, where c is a constant which can be determined by an additional constraint on $g(E_i)$'s; for example, one of $g(E_i)$'s is equal to a known number.

This theorem implies that the ratios of $g(E_i)$'s can be estimated correctly as $\delta_s \to 0$ and $k \to \infty$. It can also help us understand the weight updating procedure intuitively: To make the recursive relationship (4) invariant with respect to the weight updating, once one weight was updated, the weights of all the above levels should also be updated proportionally. Of course, this only happens for $\rho > 0$. The convergence in Eq. (4) can be intuitively argued as follows. If the $\hat{g}$ converges to a distribution, the visiting frequency to each subregion must be proportional to

$$\frac{\int_{E_i} \psi(\boldsymbol{x}) d\boldsymbol{x}}{g(E_i)}, \quad \text{for } i = 1,...,m, \tag{5}$$

because of the acceptance rule (3). By the uniqueness of the stationary distribution of the Markov chain,[19] we know as $\delta_s = 0$, Eq. (4) is the only solution for $\hat{g}(E_i)$ to converge to such that the simulation will result in the visiting frequency (5). The convergence can be checked on line in a single run. For example, we can define the statistic

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

to measure the stability of the histogram, where b is the batch size and $\hat{P}_{i,kb}$ is the normalized visiting frequency to subregion E_i calculated at the (kb)th iteration of a stage. Note that we define $0/0 = 1$ in S_k to accommodate the empty E_i's. The E_i's are set according to our knowledge of the energy function prior to the run, so probably we may overset the maximum or underset the minimum values of $H(\mathbf{x})$. This will cause some empty subregions.

The decreasing speed of δ_s may affect the convergence of the algorithm. As we know, the preceding stage simulation aims at providing a good estimate of $\hat{g}(E_i)$'s for the following stage simulation, or in other words, the following stage simulation aims at making a fine-tuning of the estimate of $\hat{g}(E_i)$'s obtained in the preceding stage. Intuitively, the δ's should be chosen such that the errors of the preceding stage estimate should be able to be corrected by the followed stage simulation with a reasonable number of iterations. Since the self-adjusting ability of the CMC moves depends on the value of δ_s, if δ_s decreases too fast, the following stage may need to take an extremely long time to correct the errors of the preceding stage estimate: otherwise, it may cause some waste of CPU time. Either case is not what we want. Hence, we suggest δ_s should decrease gradually. Our experience shows that the geometric scheme $\delta_{s+1} = \gamma \delta_s$ (with $\gamma < 1$) or the log-geometric scheme $\delta_{s+1} = \sqrt{1 + \delta_s} - 1$ works well for a variety of problems.

Following from Eqs. (4) and (5), we know that if $\rho = 0$, the resulting visiting frequency to each subregion will be

$$P_{\rho=0}(E_i) \propto \text{const}, \quad i = 1,...,m.$$

Hence, the CMC algorithm will result in a free random walk in the space of subregions. But within the same subregion, $\psi(\boldsymbol{x}) I(\boldsymbol{x} \in E_i)$ will be simulated from by the Metropolis–Hastings algorithm,[20] where $I(\cdot)$ is the indicator function. So the CMC algorithm can be regarded as a generalization of the Wang–Landau algorithm[13] to the continuous system. From the viewpoint of importance sampling, the CMC algorithm samples from the trial density

$$\pi(\boldsymbol{x}) = \sum_{i=1}^{m} \frac{\psi(\boldsymbol{x})}{g(E_i)} I(\boldsymbol{x} \in E_i),$$

as $\delta_s \to 0$. If the phase space is partitioned according to the energy function, it is easy to imagine that $\pi(x)$ is actually a function defined on a contour plot; a different weight g_i associates with a different energy level. In this sense, we call the algorithm a contour-based Monte Carlo algorithm. If we let $\psi(\boldsymbol{x}) = \exp\{-H(\boldsymbol{x})/\tau\}$, the importance weight will be

$$w(\boldsymbol{x}) = \frac{f(\boldsymbol{x})}{\pi(\boldsymbol{x})} = \sum_{i=1}^{m} \frac{g(E_i)}{Z_\tau} I(\boldsymbol{x} \in E_i)$$

$$\propto \sum_{i=1}^{m} g(E_i) I(\boldsymbol{x} \in E_i),$$

which is bounded above by $\max_i g(E_i) < \infty$.

If $\rho > 0$, the CMC algorithm will also result in 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 we want to sample from most intensively is coded as E_1, and then E_2, E_3, and so on, a choice $\rho > 0$ will result in an efficient run. This typically happens in Monte Carlo optimization runs, where we set $\rho > 0$ to bias the random walk to the low-energy region.

The CMC algorithm is so general that it includes several other algorithms as special cases. If we let $\psi(\boldsymbol{x}) \equiv 1$, $\mathcal{X}$ is finite, $T(\cdot \to \cdot)$ is a symmetric function, and $\rho = 0$, the algorithm reduces to the Wang–Landau algorithm.[13] In this case $\hat{g}$ estimates the density of states of the system. If we let $\psi(\boldsymbol{x}) \equiv 1$, $\mathcal{X}$ is finite, and $\rho = 1$, the algorithm gives a Wang–Landau-style implementation for the $1/k$-ensemble algorithm.[14] In this case $\hat{g}$ estimates the cumulative density of states of the system.

Since now we are only interested in minimizing $H(\boldsymbol{x})$, we propose the following space annealing version of the CMC algorithm to accelerate the optimization process. Suppose the phase space has been partitioned according to the energy function into m subregions $E_1,\ldots,E_m$, where E_i's are arranged in ascending order by energy. Let I_z denote the index of the subregion to which a sample x with energy $H(\boldsymbol{x}) = z$ belongs. Let $M^{(s,k)}$ denote the number of subregions we search from at the kth iteration of the sth stage of the simulation; that is, we search from $\cup_{i=1}^{M^{(s,k)}} E_i$ at the kth iteration of the sth stage of the simulation. The $M^{(s,k)}$ starts with $M^{(1,1)} = m$ and then evolves as

$$M^{(s,k)} = I_{H_{\min} + \Delta},$$

where $H_{\min}$ is the minimum energy value sampled so far in the run and Δ is a user set parameter which controls the search space of each iteration. Since $H_{\min}$ decreases monotonically, the search space shrinks iteration by iteration. In this sense we call the modified CMC algorithm the annealing CMC algorithm. Of course, the performance of the ACMC algorithm depends on the value of Δ. If Δ is too large, the algorithm may take a long time to locate the global mini-

TABLE I. Comparison of the ACMC algorithm with the pruned-enriched-Rosenbluth method (PERM) and the conventional Metropolis Monte Carlo method for the 2D AB models. The sequences tested are as follows: 13-mer: ABBABBABABBAB; 21-mer: BABABBABABBABBABABBAB; 34-mer: ABBABBABABBABBABABABBABBABABBABBAB; 55-mer: BABABBABABBABBABABBABABBABBABABBABBABABBABBAB.

				ACMC		
N	Putative ground energy[a]	PERM[b]	Metropolis[c]	Average[d]	Best[e]	Post[f]
13	−3.2939	−3.2167	−3.2235	−3.2743 (.0063)	−3.2892	−3.2941
21	−6.1976	−5.7501	−5.2881	−6.1573 (.0032)	−6.1689	−6.1979
34	−10.7001	−9.2195	−8.9749	−9.8389 (.1325)	−10.7556	−10.8060
55	−18.5154	−14.9050	−14.4089	−16.8572 (.0737)	−17.9900	−18.7407

[a]The putative ground energy value is reported in Ref. 7. They are obtained by the conjugate gradient method with initial configurations sampled by PERM.
[b]The minimum energy value sampled by PERM in all runs in Ref. 7.
[c]The minimum energy value sampled by the conventional Metropolis Monte Carlo in all runs in Ref. 11.
[d]The averaged minimum energy value sampled by the ACMC algorithm and the standard error of the average. They were computed over 20 runs for the 13-mer, 21-mer, and 34-mer sequences and over 50 runs for the 55-mer sequence.
[e]The minimum energy sampled by the ACMC algorithm in all the runs.
[f]The minimum energy obtained by the post Metropolis moves.

mum. If Δ is too small, the algorithm may miss the global minimum forever if our proposed distribution is not very spread. We note that a similar idea has appeared in Ref. 21 in applying the multicanonical algorithm[22] to traveling salesman problems.

IV. NUMERICAL RESULTS

In this paper we restricted ourselves to the AB models with Fibonacci sequences studied in Refs. 7 and 11. The Fibonacci sequence is defined recursively by

$$S_0=A, \quad S_1=B, \quad S_{i+1}=S_{i-1}\oplus S_i,$$

where $\oplus$ is the concatenation operator. The lengths of the sequences are given by the Fibonacci numbers $N_{i+1}=N_{i-1}+N_i$. Following Ref. 7, we considered sequences with lengths 13, 21, 34, and 55 in this paper. For sequences with length less than 13, our minimum energies agree perfectly with that reported in Ref. 11.

First we consider the 2D AB model. For the 13-mer sequence, we partitioned the phase space into $E_1,\ldots,E_{201}$ with an equal energy bandwidth of 0.1; that is, we set $E_1=\{\boldsymbol{x}\in\mathcal{X}:H(\boldsymbol{x})\leqslant -19.9\}$, $E_2=\{\boldsymbol{x}\in\mathcal{X}:-19.9<H(\boldsymbol{x})\leqslant -19.8\},\ldots$, and $E_{201}=\{x\in\mathcal{X}:H(\boldsymbol{x})>0.0\}$. Later, we realize that $E_1,\ldots,E_{167}$ are all empty sets, as the putative ground energy we found is -3.2941. In simulations, we set $\rho=1$, $\Delta=5$, $n_1=2.5e+6$, and $n_{s+1}=1.1n_s$, where n_s denotes the number of iterations performed in the sth stage of the simulation. The simulation proceeds until $\delta_s<0.01$. Here we set a high terminal value for δ, as our target is minimizing $H(\boldsymbol{x})$, instead of simulating from $f(\boldsymbol{x})$. We had three types of local moves (described below), which happen equally likely at each iteration. Let $\boldsymbol{x}_k=(\theta_2,\ldots,\theta_{N-1})$ denote the current state of the inhomogeneous Markov chain. In the type-I move, a component of x_k is picked up at random to undergo a modification by a Gaussian random variable $\epsilon\sim N(0,s^2[H(\boldsymbol{x}_k)-H_0])$, where s is a user tunable parameter and H_0 is a user guessed lower bound for $H(\boldsymbol{x}_k)$. We set $H_0=-20$ for all 2D cases considered in the paper. The variance of ϵ suggests a different step size $s\sqrt{H(\boldsymbol{x}_k)-H_0}$ for different states. For high-energy states, the step size is large, and for low-energy states, the step size is small. This allows the sampler to move through the high-energy region fast and explore the low-energy region in detail. The type-II move is the same with the type-I move, except for which two components are picked up at random to undergo the modifications. In the type-III move, a spherical proposal distribution is used: A direction is generated uniformly, and then the radius is drawn from $N(0,2s^2[H(\boldsymbol{x}_k)-H_0])$. The parameter s is calibrated such that the overall acceptance rate of the three types of moves is about 0.2, as suggested by Gelman *et al.*[23] For the 13-mer sequence, we set $s=0.15$.

As ACMC results in a random walk among subregions, $E_1,\ldots,E_{M(s,k)}$, it may be easy for the Markov chain to reach the attraction basin of the global minimum, but it may not be easy for it to locate the global minimum exactly. So we suggest the minimum energy configuration sampled by the ACMC algorithm be further subject to a post-minimization procedure—say, conjugate gradient minimization or just a few hundred steps of Metropolis moves[20] at a very low temperature. In this paper, the Metropolis moves were implemented with temperature $t=1.0e-7$. For the 13-mer sequence, the ACMC algorithm was run for 20 times independently. The CPU time cost by each run is bout 15 min on a 2.8 GHz computer (all computations reported in this paper were run on this computer). The computational results are summarized in Table I together with those for the other sequences.

For the 21-mer and 34-mer sequences, we had the exactly same parameter setting as for the 13-mer sequence. For the 21-mer sequence, the ACMC algorithm was run 20 times independently. Each run costs about 36 min CPU time. For the 34-mer sequence, the ACMC algorithm was also run 20 times independently. Each run costs about 76 min CPU time. For the 55-mer sequence, we partitioned the phase space with an equal energy bandwidth of 0.2, set $\Delta=10$ and $s=0.1$, and kept the setting of other parameters the same with

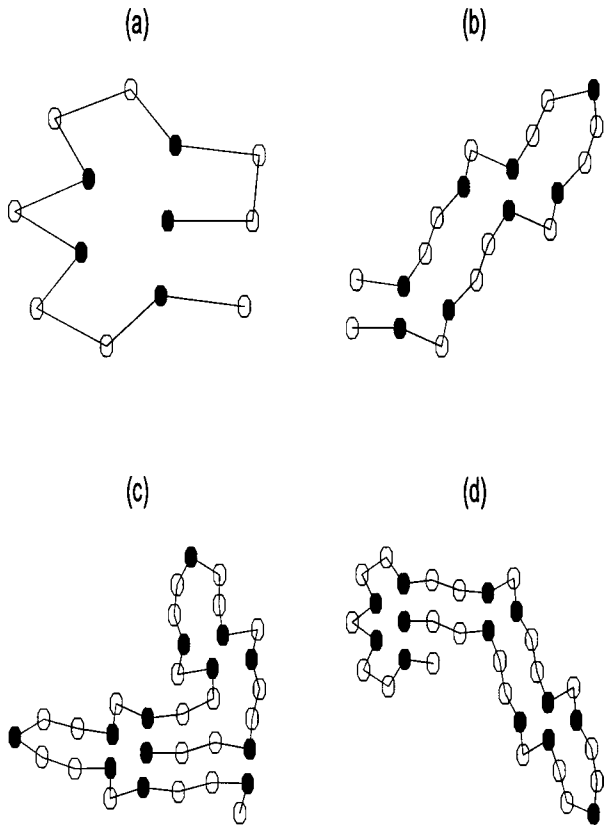

FIG. 1. Local minimum energy configurations obtained by the ACMC algorithm (subject to post Metropolis moves) for the 13-mer, 21-mer, and 34-mer sequences in the 2D AB model. The solid and open circles indicate the hydrophobic and hydrophilic monomers, respectively. (a) A configuration of the 13-mer protein with energy -3.2941. (b) A configuration of the 21-mer protein with energy -6.1979. (c) A configuration of the 34-mer protein with energy -10.8060. (d) A configuration of the 34-mer protein with energy -10.5934.

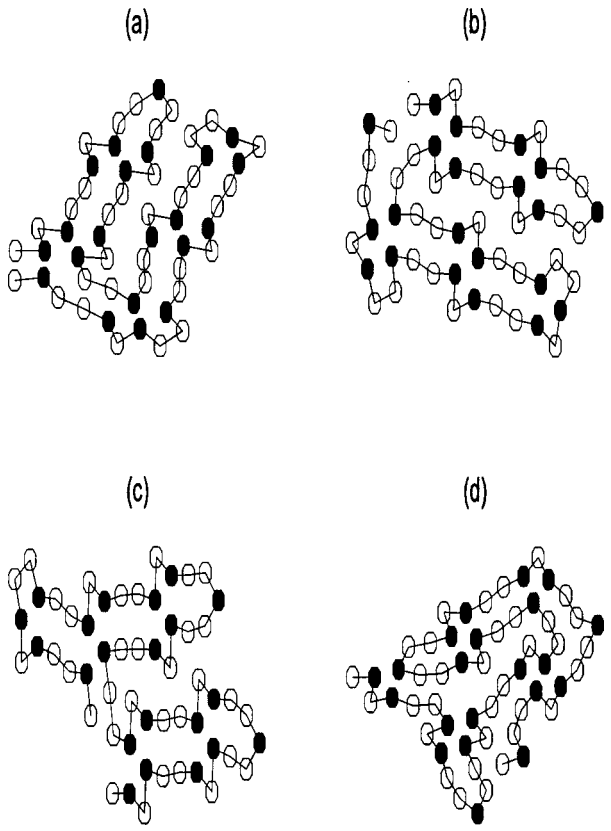

FIG. 2. Local minimum energy configurations obtained by the ACMC algorithm (subject to post Metropolis moves) for the 55-mer sequence in the 2D AB model. The solid and open circles indicate the hydrophobic and hydrophilic monomers, respectively. (a) A configuration with energy -18.7407. (b) A configuration with energy -18.5645. (c) A configuration with energy -18.2219. (d) A configuration with energy -18.2028.

that used for the 13-mer sequence. The ACMC algorithm was run for 50 times independently. Each run costs about 180 min CPU time. For comparison, the results appearing in the literature for the 2D AB model are also given in Table I. The comparison shows that the ACMC algorithm has made a significant improvement over the pruned-enriched-Rosenbluth method (PERM) and the conventional Metropolis Monte Carlo method in locating the putative ground states for the 2D AB model. For all four sequences, the averaged minimum energy sampled by the ACMC algorithm is better than the minimum energy sampled by the PERM and Metropolis Monte Carlo methods in all runs. Also, the ACMC algorithm has renewed the putative ground energies for all four sequences. Note that the 13-mer and 21-mer putative ground configurations we found are almost the same with that found by the PERM, except for some folding angles. The differences of the energy values come from the minor differences of the folding angles. The CPU times used by the ACMC algorithm should be less than or comparable to that used by the PERM. Reference 7 did not give the exact CPU times of their runs. They just mentioned that their results were obtained on their Unix or Linux workstations with up to 2 days CPU time.

Figure 1 shows some of the local minimum energy configurations found by the ACMC algorithm (subject to post Metropolis moves) for the 13-mer, 21-mer, and 34-mer sequences. The two 34-mer configurations reported here are both different from the one reported in Ref. 7. Figure 2 shows four of the local minimum energy configurations found by the ACMC algorithm for the 55-mer sequence. The four configurations are all different from the one reported in Ref. 7, but configuration (b) is quite similar to it. From Figs. 1 and 2, it is easy to see that the hydrophobic (A) monomers tend to form a hydrophobic core (in the 13-mer sequence) or clusters of typically 4–5 monomers (in other sequences) in the 2D AB model. This can be explained by the fact that hydrophobic monomers are always flanked by the hydrophilic monomers along the sequence.

The ACMC algorithm was also applied to the 3D AB model. For the 13-mer sequence, we partitioned the phase space with an equal energy bandwidth of 1.0 and set $\Delta=25$, $n_1=2.5e+6$, $n_{s+1}=1.1n_s$, $H_0=-30$, and $s=0.05$. The ACMC algorithm was run for 20 times independently. The CPU time cost by each run was about 20 min. The computational results are summarized in Table II together with the results for the other three sequences. For the 21-mer and 34-mer sequences, we used the exactly same setting as that used for the 13-mer sequence, except that we set $H_0=-60$ and $H_0=-100$, respectively. For the 55-mer sequence, we partitioned the phase space with an equal energy bandwidth

TABLE II. Computational results of the ACMC algorithm for the 3D AB model.

N	Average[a]	Best[b]	Post[c]
13	−25.7385 (.0755)	−26.3632	−26.5066
21	−49.1333 (.0937)	−50.8601	−51.7175
34	−86.2818 (.3473)	−92.7458	−94.0431
55	−137.4785 (.6884)	−149.4810	−154.5050

[a]The averaged minimum energy sampled by the ACMC algorithm and the standard error of the average. They were computed over 20 runs for the 13-mer sequence and 50 runs for the other sequences.
[b]The minimum energy sampled by the ACMC algorithm in all the runs.
[c]The minimum energy found by the post Metropolis moves.

of 2.0, set $\Delta=50$ and $H_0=-160$, and kept the setting of the other parameters the same with that used for the 13-mer sequence. The ACMC algorithm was run 50 times for each of the three long sequences. The CPU times cost by each run were about 45, 93, and 216 min, respectively. Since there exist no published ground-state energies for the 3D AB model with energy function (2) in the literature, we are unable to compare the ACMC algorithm with other methods. We note that Ref. 7 also considered the 3D AB model, but with energy function (1). The resulting local minimum energy configurations are not very realistic compared to those we present below.

Figures 3–6 show some of the local minimum energy configurations found by the ACMC algorithm (subject to post Metropolis moves) for the four sequences. We can see that all four sequences fold into compact globular structures with single hydrophobic cores. Compared to the 2D AB model, the 3D AB model is more realistic for Fibonacci sequences.

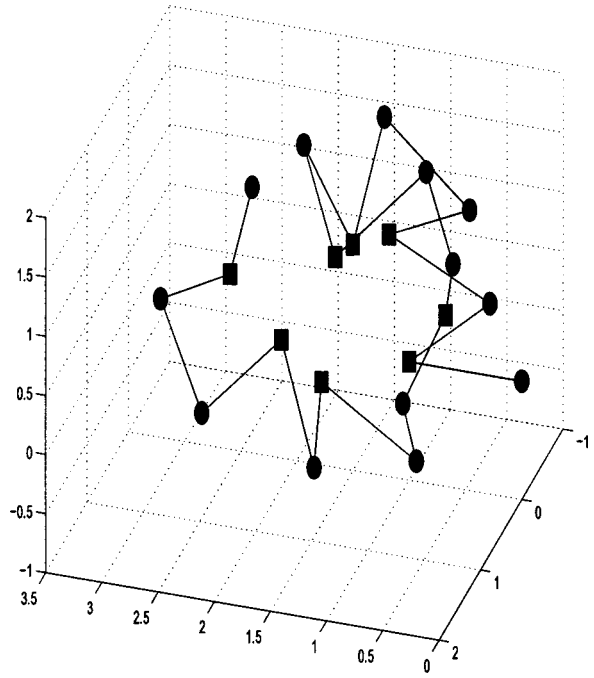

FIG. 4. A local minimum energy configuration of the 21-mer sequence with energy −51.7175. The square and dot indicate the hydrophobic and hydrophilic monomers, respectively.

V. DISCUSSION

We have applied successfully the ACMC algorithm to the 2D and 3D AB models. The numerical results show that the ACMC algorithm is a very promising algorithm for a general optimization task. Note that in the simulation we did not use any moves or proposal functions which are specific to AB models. In other words, if we can incorporate some

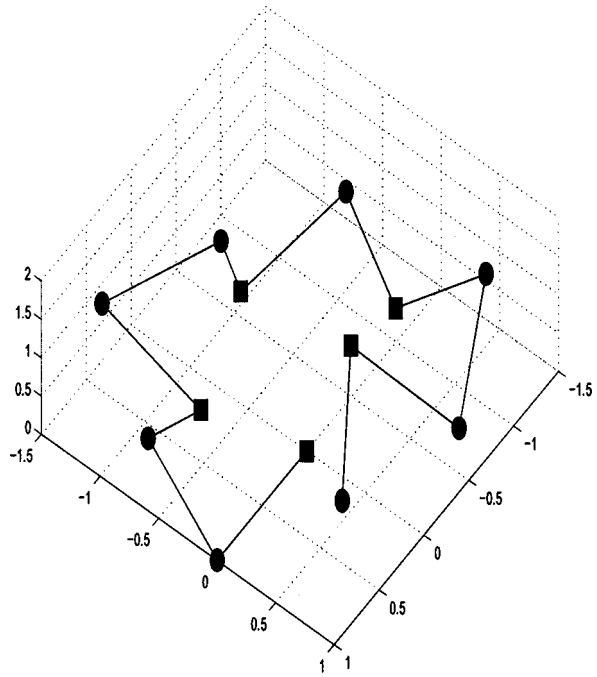

FIG. 3. A local minimum energy configuration of the 13-mer sequence with energy −26.5066. The square and dot indicate the hydrophobic and hydrophilic monomers, respectively.

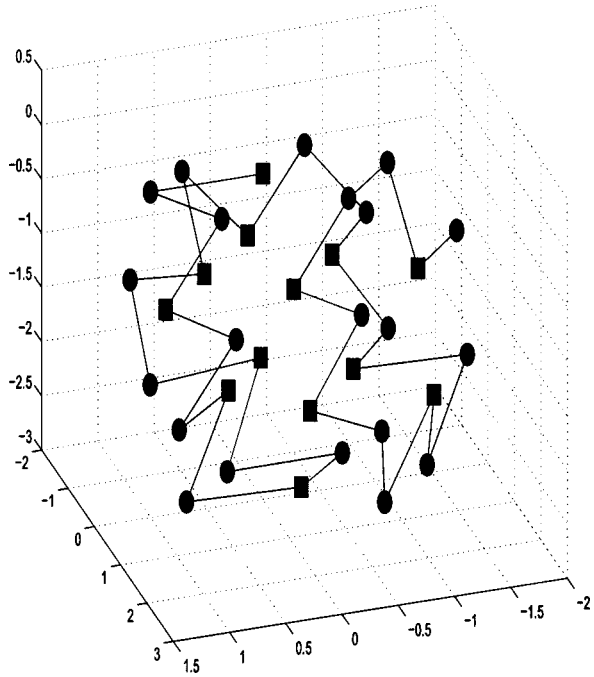

FIG. 5. A local minimum energy configuration of the 34-mer sequence with energy −94.0431. The square and dot indicate the hydrophobic and hydrophilic monomers, respectively.

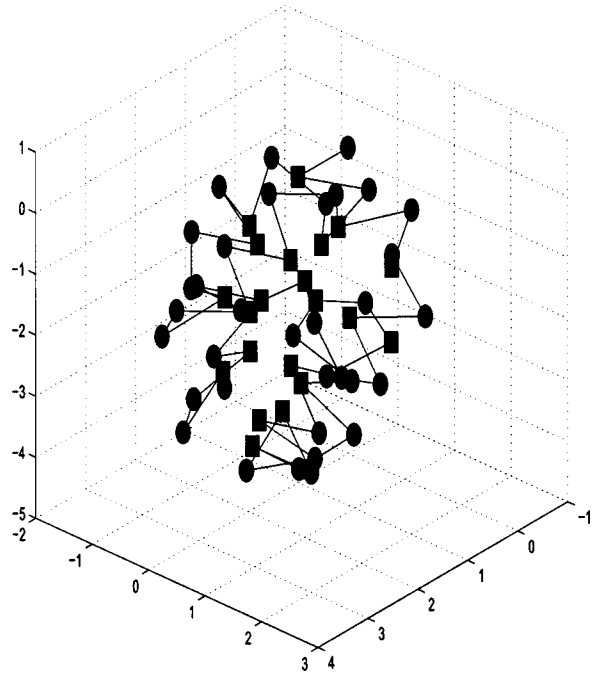

FIG. 6. A local minimum energy configuration of the 55-mer sequence with energy −154.5050. The square and dot indicate the hydrophobic and hydrophilic monomers, respectively.

specific moves, which are designed based on some specific properties of the protein model, into the ACMC algorithm, the performance of the ACMC algorithm may be further improved. The success of the ACMC algorithm in the AB models suggests that the ACMC algorithm may also perform well for all-atom models with realistic potentials, at least for some small proteins.

About the effect of the parameter ρ in the ACMC algorithm, we have the following comments. In this paper we set $\rho=1$ in all simulations. This is not necessary. In fact, our simulations indicate that a setting of ρ ranging from 0.5 to 1.5 will produce almost the same results. However, a setting of $\rho=0$ or $\rho>3$ usually performs less well, especially for long sequences. The reason is obvious. The setting of $\rho=0$ will result in a free random walk among all subregions, and hence, the simulation will waste too much time in the high-energy region, while the setting with a large value of ρ will bias the simulation too much to the low-energy region, and hence, the simulation will tend to get trapped in a local energy minimum. This is also consistent with the result reported in Ref. 14, where the author claimed that the $1/k$-ensemble algorithm is more efficient than the multicanonical algorithm in estimating the density of states for a discrete system.

At last, we would like to mention one potential application of the CMC algorithm. It can be used to estimate the partition function of a model. Although it is not the focus of this paper, it may be of interest to physicists or chemists, as the behavior of a model can be described by its partition function. For example, if we set $\rho=0$ and $\psi(\boldsymbol{x})=\exp\{-H(\boldsymbol{x})/\tau\}$ and assume that $g(E_m)=\int_{E_m}\exp\{-H(\boldsymbol{x})/\tau\}\boldsymbol{x}$ has been estimated *a priori*, then the partition function of the model can be estimated by

$$\hat{Z}=\sum_{i=1}^{m}\hat{g}(E_i).$$

The $g(E_m)$ is the partition function of the distribution truncated on the highest-energy subregion E_m, and it can be estimated easily using the conventional Monte Carlo method, as the simulation is restricted only to the highest-energy region. In a CMC run, if we confine $\hat{g}(E_m)=\hat{g}^{(0)}(E_m)$, the estimate of $g(E_m)$ obtained *a priori*, then the other estimates $\hat{g}(E_i)$'s can be determined according to the theorem. This use of the CMC algorithm will be further explored elsewhere.

APPENDIX: PROOF OF THE THEOREM

For simplicity, in the proof we will denote $g(E_i)$ by g_i and denote $\hat{g}^{(s,k)}(E_i)$ by $\hat{g}_i^{(k)}$ by omitting the superscript s as the calculation is only for one stage. Let $D_k=\Sigma_{i=1}^{m}|\hat{g}_i^{(k)}-g_i|$. Without loss of generality, we assume that $S=\Sigma_{i=1}^{m}g_i$ is known, $\Sigma_{i=1}^{m}\hat{g}_i^{(k)}=S$, and the sample of the next iteration, $\boldsymbol{x}_{k+1}\in E_j$. To make $\Sigma_{i=1}^{m}\hat{g}_i^{(k+1)}=S$, we set

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

where $\tau_j=\epsilon_j\hat{g}_j^{(k)}/S$, where $\epsilon_j=\delta_s(1-\rho^{m-j+1})/(1-\rho)$ for $\rho\neq1$ and $\epsilon_j=(m-j+1)\delta_s$ for $\rho=1$. The weight adjustment by the multiplication factor $(1+\tau_j)$ will not change the process of the simulation. For D_{k+1}, we have the calculation

$$\begin{aligned}E(D_{k+1}|\hat{g}_i^{(k)},i&=1,\ldots,m)=\sum_{j=1}^{m}\left\{\sum_{i=1}^{j-1}\left|\frac{\hat{g}_i^{(k)}}{1+\tau_j}-g_i\right|\right.\\&\left.+\sum_{i=j}^{m}\left|\frac{\hat{g}_i^{(k)}+\delta_s\rho^{i-j}\hat{g}_j^{(k)}}{1+\tau_j}-g_i\right|\right\}\\&\times P(\boldsymbol{x}_{k+1}\in E_j)=\sum_{j=1}^{m}\left\{\sum_{i=1}^{m}\left|\frac{\hat{g}_i^{(k)}}{1+\tau_j}-g_i\right|\right\}\\&\times P(\boldsymbol{x}_{k+1}\in E_j)\\&+\sum_{j=1}^{m}\left\{\sum_{i=j}^{m}\left[\left|\frac{\hat{g}_i^{(k)}+\delta_s\rho^{i-j}\hat{g}_j^{(k)}}{1+\tau_j}-g_i\right|\right.\right.\\&\left.\left.-\left|\frac{\hat{g}_i^{(k)}}{1+\tau_j}-g_i\right|\right]\right\}P(\boldsymbol{x}_{k+1}\in E_j)=(\mathrm{I})+(\mathrm{II}).\end{aligned}$$

Given $\hat{g}_i^{(k)}$, $i=1,\ldots,m$, the acceptance of a CMC move is guided by the Metropolis–Hastings rule, so we have

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

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

$$
\begin{aligned}
(\mathrm{I}) &\leqslant \sum_{j=1}^{m} \left\{ \frac{1}{1+\tau_j} \sum_{i=1}^{m} \left| \hat{g}_i^{(k)} - g_i \right| + \frac{\tau_j}{1+\tau_j} \sum_{i=1}^{m} g_i \right\} P(\boldsymbol{x}_{k+1} \in E_j) \\
&\leqslant \frac{1}{1+\tau_0} D_k + \sum_{j=1}^{m} \frac{\epsilon_j \hat{g}_j^{(k)}}{1+\tau_j} P(\boldsymbol{x}_{k+1} \in E_j) \\
&\quad \times (\text{use the relationship } \epsilon_j \hat{g}_j^{(k)} \\
&= \tau_j S) \\
&\leqslant \frac{1}{1+\tau_0} D_k + \sum_{j=1}^{m} \frac{\epsilon_j \int_{E_j} \psi(\boldsymbol{x}) d\boldsymbol{x}}{A(1+\tau_j)} \\
&\leqslant \frac{1}{1+\tau_0} D_k + \frac{\delta_s G}{(1+\tau_0)A} \sum_{i=1}^{m} \rho^{i-1},
\end{aligned}
$$

where $\tau_0 = \min_{1 \leqslant j \leqslant m} \tau_j$ and $G = \int_{\mathcal{X}} \psi(\boldsymbol{x}) d\boldsymbol{x} < \infty$. Similarly, we have

$$
\begin{aligned}
(\mathrm{II}) &\leqslant \sum_{j=1}^{m} \left\{ \sum_{i=j}^{m} \frac{\delta_s \rho^{i-j} \hat{g}_j^{(k)}}{1+\tau_j} \right\} P(\boldsymbol{x}_{k+1} \in E_j) \\
&\leqslant \sum_{j=1}^{m} \frac{\epsilon_j \hat{g}_j^{(k)}}{1+\tau_j} P(\boldsymbol{x}_{k+1} \in E_j) \leqslant \frac{\delta_s G}{(1+\tau_0)A} \sum_{i=1}^{m} \rho^{i-1}.
\end{aligned}
$$

From (I) and (II), we have

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

It implies that

$$
E(D_{k+1}) \leqslant \frac{1}{1+\tau_0} E(D_k) + O(\delta_s)
$$

and

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

Since $D_k > 0$, we know

$$
D_k \to 0 \quad \text{in probability}
$$

and then

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

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

[1] K. F. Lau and K. A. Dill, Macromolecules **22**, 3986 (1989); H. S. Chan and K. A. Dill, J. Chem. Phys. **95**, 3775 (1991); D. Shortle, H. S. Chan, and K. A. Dill, Protein Sci. **1**, 201 (1992).
[2] K. Yue and K. A. Dill, Phys. Rev. E **48**, 2267 (1993); Proc. Natl. Acad. Sci. U.S.A. **92**, 146 (1995).
[3] T. C. Beutler and K. A. Dill, Protein Sci. **5**, 2037 (1996).
[4] G. Chikenji, M. Kikuchi, and Y. Iba, Phys. Rev. Lett. **83**, 1886 (1999); G. Chikenji and M. Kikuchi, Proc. Natl. Acad. Sci. U.S.A. **97**, 14273 (2000).
[5] F. Liang and W. H. Wong, J. Chem. Phys. **115**, 3374 (2001).
[6] J. L. Zhang and J. S. Liu, J. Chem. Phys. **117**, 3492 (2002).
[7] H. P. Hsu, V. Mehra, W. Nadler, and P. Grassberger, J. Chem. Phys. **118**, 444 (2003); Phys. Rev. E **68**, 021113 (2003).
[8] S. Will, in *Proceedings of the Pacific Symposium on Biocomputing 2002 (PSB 2002)*, edited by Russ B. Altman *et al.* (World Scientific, Singapore, 2002); R. Backofen and S. Will, in *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching (CPM2001)*, edited by Amihood Amir *et al.*, Lecture Notes in Computer Sciences 2089 (Springer-Verlag, Berlin, 2001); for a web application, see http://www.bio.inf.uni-jena.de/Prediction/prediction.cgi
[9] H. P. Hsu, V. Mehra, and P. Grassberger, Phys. Rev. E **68**, 037703 (2003).
[10] F. H. Stillinger, T. Head-Gordon, and C. L. Hirshfeld, Phys. Rev. E **48**, 1469 (1993); T. Head-Gordon and F. H. Stillinger, *ibid.* **48**, 1502 (1993).
[11] F. H. Stillinger and T. Head-Gordon, Phys. Rev. E **52**, 2872 (1995).
[12] F. Liang (unpublished).
[13] F. Wang and D. P. Landau, Phys. Rev. Lett. **86**, 2050 (2001); Phys. Rev. E **64**, 056101 (2001).
[14] B. Hesselbo and R. B. Stinchcombe, Phys. Rev. Lett. **74**, 2151 (1995).
[15] A. Irbäck, C. Peterson, and F. Potthast, Phys. Rev. E **55**, 860 (1997).
[16] A. Irbäck and F. Potthast, J. Chem. Phys. **103**, 10298 (1995).
[17] A. Irbäck, C. Peterson, F. Potthast, and O. Sommelius, J. Chem. Phys. **107**, 273 (1997).
[18] D. Gorse, Biopolymers **64**, 146 (2002).
[19] L. Tierney, Ann. Stat. **22**, 1701 (1994).
[20] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953); W. K. Hastings, Biometrika **57**, 97 (1970).
[21] J. Lee and M. Y. Cho, Phys. Rev. E **50**, 651 (1994).
[22] B. A. Berg and T. Neuhaus, Phys. Lett. B **267**, 291 (1991); Phys. Rev. Lett. **68**, 9 (1992); B. A. Berg and T. Celik, *ibid.* **69**, 2292 (1992).
[23] A. Gelman, R. O. Roberts, and W. R. Gilks, in *Bayesian Statistics 5*, edited by J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith (Oxford University Press, Oxford, 1996).