The Korean Journal of Applied Statistics (2009)
22(3), 627–634

Real Protein Prediction in an Off-Lattice BLN Model via Annealing Contour Monte Carlo

Sooyoung Cheon[1]

[1]KU Industry-Academy Cooperation Group Team of Economics and Statistics, Korea University

(Received February 2009; accepted April 2009)

Abstract

Recently, the general contour Monte Carlo has been proposed by Liang (2004) as a space annealing version(ACMC) for optimization problems. The algorithm can be applied successfully to determine the ground configurations for the prediction of protein folding. In this approach, we use the distances between the consecutive C_α atoms along the peptide chain and the mapping sequences between the 20-letter amino acids and a coarse-grained three-letter code. The algorithm was tested on the real proteins. The comparison showed that the algorithm made a significant improvement over the simulated annealing(SA) and the Metropolis Monte Carlo method in determining the ground configurations.

Keywords: Protein folding, annealing contour Monte Carlo, simulated annealing, BLN model.

1. Introduction

Prediction of tertiary structure of proteins or protein folding from its sequence has been one of the most challenging problems in biophysics. The difficulty of the problem is that the dimension of the system is usually high because it is in the same order with the number of atoms involved in the system, and the energy landscape can be characterized by a multitude of local minima separated by high-energy barriers. At low temperatures, the conventional Markov chain Monte Carlo algorithms, such as the Metropolis-Hastings algorithm(MH; Metropolis *et al.*, 1953; Hastings, 1970), and molecular dynamic simulations tend to get trapped in local minima. Thus it makes the simulations ineffective. Attempts to alleviate this problem have been developed and successfully applied to various problems such as Monte Carlo with minimization (Li and Scheraga, 1987), simulated annealing(SA; Kirkpatrick *et al.*, 1983) and genetic algorithms(GA; Holland, 1975; Goldberg, 1989). Furthermore, there has been studied in finding the global minimum-energy conformation by studying the off-lattice model protein, so-called BLN model, introduced by Honeycutt and Thirumalai to avoid the high energy barriers between local minima in recent years (Kim *et al.*, 2003). The BLN model is a coarse-grained model that uses an α-carbon trace to represent the protein backbone in which structural details of the amino acids and aqueous solvent have been integrated out and replaced with effective bead-bead interactions. This model is particularly useful in the study of predicting the native structure of a protein.

[1]Research Professor, KU Industry-Academy Cooperation Group Team of Economics and Statistics, Korea University, Jochiwon 339-700, Korea. E-mail: scheon@korea.ac.kr

Table 2.1. Sequence mapping between 20-letter(20) amino acid and coarse-grained three-letter(3) code

20	3	20	3	20	3	20	3
Ala	B	Met	B	Gly	N	Asn	L
Cys	B	Val	B	Ser	N	His	L
Leu	B	Trp	B	Thr	L	Gln	L
Ile	B	Tyr	B	Glu	L	Lys	L
Phe	B	Pro	N	Asp	L	Arg	L

Recently, Kim *et al.* (2005) proposed the conformational space annealing(CSA) algorithm that generates lower ground-state energies than those reported in the literature in off-lattice AB protein model. Zhang *et al.* (2007) suggested that the genetic-annealing algorithm is very promising for searching the ground states of protein folding in three dimensional off-lattice AB model. However, they didn't investigate real proteins in Protein Data Bank(PDB).

In this paper, we apply the ACMC algorithm to the real proteins with the BLN model. The contour Monte Carlo(CMC) algorithm (Liang, 2005) is a generalization of the Wang-Landau algorithm (Wang and Landau, 2001) and the $1/k$-ensemble algorithm (Hesselbo and Stinchcombe, 1995). 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. The ACMC algorithm is an accelerated version of the CMC algorithm for optimization problems. Our numerical results show that ACMC is a very promising algorithm for finding the ground configurations of proteins.

The remaining part of this paper is organized as follows. In Section 2, we describe the BLN model. In Section 3, we review the ACMC algorithm briefly. Section 4 presents numerical results for real small proteins. In Section 5, we conclude the paper.

2. The Model

The energy function for the model is proposed in the off-lattice frustrated model developed by Kim *et al.* (2003). The protein chain is modeled as a sequence of beads of three flavors: hydrophilic, hydrophobic, and neutral; these are designated by L, B and N, respectively(Table 2.1). Attraction between the hydrophobic beads provides the energetic driving force for formation of a strong core. Repulsion between the hydrophilic beads and other beads is used to balance the forces and reduce the bias of the correct native fold. The neutral beads also serve with little repulsion and typically signal the turn regions in the sequence.

The total potential energy function of the BLN model with M residues is given by

$$\begin{aligned} V &= V_b + V_a + V_t + V_v \\ &= \sum_{i=1}^{M-1} \frac{K_r}{2}(|r_{i+1} - r_i| - a)^2 + \sum_{i=1}^{M-2} \frac{K_\theta}{2}(\theta_i - \theta_0)^2 + \sum_{i=1}^{M-3} \{A_i(1+\cos\phi_i) + B_i(1+\cos 3\phi_i)\} + \\ &\quad 4\epsilon \sum_{i=1}^{M-3} \sum_{j=i+3}^{M} C_{ij} \left\{ \left(\frac{\sigma}{r_{ij}}\right)^{12} - D_{ij} \left(\frac{\sigma}{r_{ij}}\right)^{6} \right\}, \end{aligned}$$

where V_b is the bond-stretching energy, V_a the bond-angle bending energy, V_t the torsional dihedral-angle energy and V_v the van der Waals energy. The force constant is given by $K_r = 400\epsilon/a^2$, ϵ is the energy constant, a the average bond length, and r_i the position of the i^{th} residue. $K_\theta = 20\epsilon/(\text{rad})^2$, θ_i is the bond angle defined by three residuals i, $i+1$ and $i+2$, and $\theta_0 = 1.8326$ rad or 105^o. ϕ_i

is the dihedral angle defined by four residues i, $i+1$, $i+2$ and $i+3$. Each dihedral in the chain is predefined: $A_i = 0$ and $B_i = 0.2\epsilon$ if two or more of the four residues are neutral; $A_i = B_i = 1.2\epsilon$ for all the other cases. σ is the Lennard-Jones parameter and r_{ij} is the distance between two nonbonded residues i and j given by $r_{ij} = |r_i - r_j|$. The nonlocal interactions are given by $C_{ij} = D_{ij} = 1$ for BB interactions, $C_{ij} = 2/3$ and $D_{ij} = -1$ for LL and LB interactions, and $C_{ij} = 1$ and $D_{ij} = 0$ for all interactions involving N residues. All simulations are performed in reduced units, with a, ϵ and σ all set equal to unity.

3. Annealing Contour Monte Carlo Algorithm

In this section we describe the space annealing version of CMC algorithm (Liang, 2004). First the general CMC algorithm is described briefly and then present its space annealing version for optimization problems. Refer to Liang (2005) for the details of CMC.

Suppose that we are interested in sampling from the following distribution,

$$f(x) = \frac{1}{Z} \exp\left\{-\frac{H(x)}{\tau}\right\}, \quad x \in \mathcal{X}, \tag{3.1}$$

where τ is called the temperature of the system and $Z = \int_{\mathcal{X}} \exp\{-H(x)/\tau\}\, dx$ is the partition function where $\mathcal{X}$ is called the sample space. The CMC algorithm works in the style of importance sampling; that is, it may work directly on any nonnegative function $\psi(x)$ defined on the phase space $\mathcal{X}$, instead on $f(x)$ necessarily, although we often set $\psi(x) = \exp\{-H(x)/\tau\}$ for convenience. For the function $\psi(x)$, we usually require that $\int_{\mathcal{X}} \psi(x)dx < \infty$.

CMC generalizes the Wang-Landau algorithm (Wang and Landau, 2001) to continuum systems. The generalization is mainly in three respects, the sample space, the working function and the estimate updating scheme. Suppose that the sample space $\mathcal{X}$ is continuous and has been partitioned according to the energy function $H(x)$, into m disjoint subregions denoted by $E_1 = \{x : H(x) < h_1\}$, $E_2 = \{x : h_1 \le H(x) < h_2\}, \ldots, E_{m-1} = \{x : h_{m-2} \le H(x) < h_{m-1}\}$ and $E_m = \{x : H(x) \ge h_{m-1}\}$, where $h_1, \ldots, h_{m-1}$ are $m-1$ specified real numbers. Let $g(E_i) = \int_{E_i} \psi(x)dx$ 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 $T(\cdot \to \cdot)$ denote a proposal distribution which is not necessarily symmetric, and $x_{s,k}$ and $\hat{g}^{(s,k)}(E_i)$ denote the sample and the estimate of $g(E_i)$, respectively, at the k^{th} iteration of the s^{th} 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 an initial random sample $x_{0,0}$, and then iterates as follows.

1. Propose a new configuration x^* in the neighborhood of $x_{s,k}$ according to a prespecified distribution $T(\cdot \to \cdot)$.
2. Accept x^* with probability

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

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

$i = 0, \ldots, m - I_{x_{s,k}}$, where $\rho > 0$ is a parameter which controls the sampling frequency for each of the subregions.

The CMC algorithm is so general that it includes several other algorithms as special cases. If we let $\psi(x) \equiv 1$, $\mathcal{X}$ is finite, $T(\cdot \to \cdot)$ is a symmetric function, and $\rho = 0$, the algorithm reduces to the Wang-Landau algorithm (Wang and Landau, 2001). In this case, $\hat{g}$ estimates the density of states of the system. If we let $\psi(x) \equiv 1$, $\mathcal{X}$ is finite, and $\rho = 1$, the algorithm gives a Wang-Landau-style implementation for the $1/k$-ensemble algorithm (Hesselbo and Stinchcombe, 1995). In this case $\hat{g}$ estimates the cumulative density of states of the system.

Since the goal is to minimize the energy function $H(x)$ in this paper, the following space annealing version of the CMC algorithm is used for accelerating the optimization process. Let I_z denote the index of the subregions searching where a sample x with energy $z = H(x)$ belongs to. Let $M^{(s,k)}$ denote the number of subregions searching from the set $\bigcup_{i=1}^{M^{(s,k)}} E_i$ at the k^{th} iteration of the s^{th} stage of the simulation. The $M^{(s,k)}$ starts with $M^{(1,1)} = m$ and 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 specified value which controls the search space of each iteration. Since H_{min} decreases monotonically, the search space shrinks iteration by iteration. The performance of ACMC also depends on the value of Δ. If Δ is too large or small, the algorithm may take a long time to locate the global minimum or miss the global minimum for ever if the proposal distribution is not very spread, respectively.

For an effective implementation of ACMC, several issues need to be considered.

- *Sample space partition.* The sample space partition should balance the following two considerations. The first one is the acceptance rate of the sampling steps. Suppose that the MH algorithm is employed to sample from the working density. Since the transition ratio of the MH moves

$$r(x \to y) = \frac{\hat{g}(E_i)}{\hat{g}(E_j)} \frac{\psi(y)}{\psi(x)} \frac{T(y \to x)}{T(x \to y)} \tag{3.2}$$

  is invertible, that is, $r(x \to y) = 1/r(y \to x)$, a partition which results in a small variance of r is preferred. For simplicity, let $T(\cdot \to \cdot)$ be symmetric, $\psi_i(x) = \psi(x)/\hat{g}(E_i)I(x \in E_i)$, and $\psi_j(y) = \psi(y)/\hat{g}(E_j)I(y \in E_j)$. We then have $r(x \to y) = \psi_j(y)/\psi_i(x)$. This suggests that to minimize the variance of r, the sample space should be partitioned such that each $\psi_i(x)$ is as uniform as possible over the corresponding subregion. We note that a partition made according to the energy function can satisfy this criterion approximately. For example, we partition the sample space into the following m subregions with an equal energy bandwidth: $E_1 = \{x : H(x) < u_1\}, E_2 = \{x : u_1 \le H(x) < u_2\}, \ldots, E_{m-1} = \{x : u_{m-2} \le H(x) < u_{m-1}\}$, and $E_m = \{x : H(x) \ge u_{m-1}\}$, where $u_i - u_{i-1} = \Delta_u$ for $i = 2, \ldots, m-1$. The u_1 and u_{m-1} can be chosen according to the problem under consideration, and Δ_u, the so-called the energy bandwidth of the partition, should be bounded by a reasonable number, say 1 or 2, to ensure that the local MH sampler can move freely within the same subregion with a reasonable overall acceptance rate. If m is too small or large, then the sample space may not be able to be well explored.

  The second one is the number of subregions. Liang (2005) suggests that a small value of m is preferred for a fast convergence. This is clear in the case of $\rho = 0$. Note this issue is in conflict with the first one. Hence, a trade-off should be made between the number of subregions and the variation of $\psi_i(x)I(x \in E_i)$'s within each subregion.

- *Number of iterations.* The iteration numbers performed in each stage can be determined

based on (A.7) in the Appendix of Liang (2005). Roughly, we should have

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

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

- *Effect of* δ. To measure the effect of δ we assume that the sampler is trapped in the current state $x \in E_i$, and a new state $y \in E_j$ is proposed with $i < j$. The transition ratio is calculated as in (3.2). Suppose that the proposal is rejected and the same proposal is made in the next iteration, the transition ratio for the new iteration will be

$$r'(x \to y) = \frac{\hat{g}(E_i) + \delta_s \hat{g}(E_i)}{\hat{g}(E_j) + \delta_s \rho^{j-i} \hat{g}(E_i)} \frac{\psi(y)}{\psi(x)} \frac{T(y \to x)}{T(x \to y)}. \tag{3.3}$$

For any ρ, it is easy to see that $r(x \to y) < r'(x \to y)$, that is, the acceptance rate is adjusted to a larger value due to the rejection of the last iteration. CMC has the ability to adjust the acceptance rate of the MH moves according to the historical samples. We call this ability the self-adjusting ability in this article. Note that the MH algorithm does not have this ability, and it tends to be trapped into a local energy minimum indefinitely. The self-adjusting ability of CMC depends on the value of δ. The larger value of δ, the higher self-adjusting ability. We usually set δ to a large value, say, 1, at early stages of the simulation. This will force the sampler to visit all subregions very fast. However, a large value of δ makes the estimates of $g(E_i)$'s inaccurate. It follows from Corollary 1 (Liang, 2005) that the accuracy of the estimates of each stage is in the order of $O(\delta)$. To improve the accuracy of the estimates $\hat{g}(E_i)$'s, δ should decrease gradually with the progress of the simulation.

Theoretically, the decreasing rate of δ will not affect the convergence of the algorithm. In this article, we suggest a geometric scheme $\delta_{s+1} = \gamma\delta_s$ for the decrease of δ_s. If we let δ_s decrease geometrically with the rate γ and let K_s increase geometrically with the rate $1/\gamma$, the total number of iterations needed for a CMC run will be approximately $K_1(\delta_1/\delta_{end})(\gamma/(1-\gamma))$. In this paper, we set $\gamma = 0.5$.

- *Effect of* ρ. Liang (2005) shows that the larger ρ, the faster convergence of the CMC algorithm. The effect of ρ can also be understood intuitively as follows. Suppose the sample space is partitioned according to the energy function, and the subregions are arranged in the ascending order of the energy. As mentioned before, a choice of $\rho > 0$ will result in that the low index subregions will be visited more frequently than the others. Since the mass of the distribution is usually dominated by low energy regions, this choice often results in an efficient simulation. However, an extremely large value of ρ may not be good. For example, if $\rho = 1$, all samples will be drawn from the subregion E_1, and the inference will be biased. However, since our aim is to minimize the energy function $H(x)$, we suggest setting ρ to a slightly large value, say, 1. This often accelerates the search process for the minimum energy solutions.

4. Numerical Results

A simple representation was adopted as the BLN model, in which a residue was reduced to a bead and its coordinate was in the position of the C_α atom of the residue. Four small proteins were selected as the tested targets from the Protein Data Bank(PDB); *i.e.*, 1bds, 1crn, 2tgp and 9ins.

The ACMC algorithm was first applied to these four real proteins. For the 1bds and 1crn proteins, we partitioned the phase space into $E_1, \ldots, E_{114}$ with an equal energy bandwidth of 2; that is, we set $E_1 = \{x \in \chi : H(x) < -158\}, E_2 = \{x \in \chi : -158 \leq H(x) < -156\}, \ldots, E_{114} = \{x \in \chi : H(x) \geq 70\}$. ACMC was run 10 times independently. In simulations, we set $\rho = 1, \Delta = 50, n_1 = 5.0e + 5$, and $n_{s+1} = 1.1n_s$, where n_s denotes the number of iterations performed in the s^{th} 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 $H(x)$, instead of simulating from $f(x)$. We utilized three types of local moves (Liang, 2004). Let $x_k = (\theta_3, \ldots, \theta_N)$ denote the current state of the inhomogeneous Markov chain. In the type-I move, a component of x_k is picked up randomly to undergo a modification by Gaussian random variable $\epsilon \sim N(0, s^2[H(x_k) - H_0])$, where s is a user tunable parameter and H_0 is a user guessed lower bound for $H(x_k)$. We set $H_0 = -160$. The variance of ϵ suggests a large(small) step size $s\sqrt{H(x_k) - H_0}$ for high(low)-energy states. 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. In the type-III move, a spherical proposal distribution is used. A direction is generated uniformly, and then the radius is drawn from $N(0, 2s^2[H(x_k) - H_0])$. 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.* (1996). For this case, we set $s = 0.15$.

As the ACMC algorithm results in a random walk among subregions, $E_1, \ldots, E_{M(s,k)}$, 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, just a few hundred steps of Metropolis moves (Gelman *et al.*, 1996) at a very low temperature. In this paper, as a post-minimization procedure, we use temperature $t = 1.0e - 7$. We set an energy bandwidth of 2.0, $\delta = 100$, $s = 0.3$ for 2tgp, and a energy bandwidth of 1.0, $\Delta = 50$, $H_0 = -100$, $n_{s+1} = 1.2n_s$, $s = 0.15$ for 9ins, and keep the same setting for all other parameters in both proteins with those used by the previous example.

For comparison, SA, the popular optimization algorithm, and the MH algorithm were also applied to these proteins. Each of the two algorithms was run 10 times independently with its default parameter setting, and each run consisted of 1.0×10^6 iterations. The computational results for four real proteins are summarized in Table 4.1. The comparison shows that the ACMC algorithm has made a significant improvement over SA and MH in locating the ground states for the BLN model based on the minimum value and root mean square deviation(rmsd). For all four proteins, the averaged minimum energies sampled by ACMC are much smaller than those by SA and MH in all runs. In particular, the MH method may tend to get trapped in the local energy minimum since its sampler never even reaches to the negative value. SA also seems to have local trap problems. The differences of the energy values come from the differences of the folding angles.

To compare the performance of ACMC according to the energy bandwidth, we run the ACMC algorithm with various energy bandwidths in 1bds protein. The computational results are summarized in Table 4.2. As we mentioned in Section 3, the sample partition should be bounded by a reasonable number, say 1 or 2, to ensure to get a reasonable overall acceptance rate, about 0.2. The comparison shows that only when the ACMC algorithm uses the bandwidth 1 or 2, it has a reasonable acceptance rate and can find the global minimum energy value. As we decrease the bandwidth, the acceptance rate decreases and the minimum energy value increases. If we increase the bandwidth, we get an opposite result in the acceptance rate. In addition, the result indicates that the simulation takes a long time if the bandwidth is very small.

Table 4.1. Comparison of the ACMC algorithm with the simulated annealing(SA) and the conventional Metropolis Monte Carlo(MH) method.

			ACMC	
N	SA(rmsd)[a] [CPU(m)]	MH(rmsd)[b] [CPU(m)]	Aver.Best[c]	Aver.Post(rmsd)[d] [CPU(m)]
9ins (30)	−41.4312(1.9983) [223.20]	18.7912(2.5317) [4.00]	−42.0567	−43.0455(1.9950) [197.78]
1bds (43)	−42.1128(2.7700) [364.27]	30.5504(3.5783) [7.30]	−47.0457	−49.6576(2.1033) [240.25]
1crn (46)	−52.6261(2.5667) [706.22]	36.5253(2.9050) [8.33]	−51.1785	−54.7721(2.1340) [634.30]
2tgp (58)	−51.5978(3.2550) [1546.57]	57.1451(3.8233) [21.17]	−48.8130	−53.3135(2.9825) [1029.30]

a: The averaged minimum energy and rmsd sampled by SA.
b: The averaged minimum energy and rmsd sampled by MH.
c: The averaged minimum energy sampled by ACMC.
d: The averaged minimum energy and rmsd sampled by the post Metropolis moves.
Averaged CPU time (in minutes) cost by ACMC on an Intel Core(TM)2 DUO computer. The values of rmsd were obtained from the each standardized structure versus the native standardized structures of the PDB, and structures of proteins tested are as follows:
1bds(43; BBNBBBNNLN NLNLBBBBLN LBNNNBNBLN LBBLBNLBBB BNL);
1crn(46; LLBBNNBBBL NLBLBBLBNN LNLBBBBLBL NBBBBNNBLB NNLBBL);
2tgp(58; LNLBBBLNNB LNNBLBLBBL BBBLBLBNBB LLBBBNNBLB LLLLBLNBLL BBLLBNNB);
9ins(30; BBLLLBBNNL BBLBBBBBBN LLNBBBLNLB).

Table 4.2. Comparison of ACMC with various energy bandwidths in 1bds.

Bandwidth	CPU(m)[a]	Acceptance Rate	Best[b]	Aver.Post[c]
0.05	726.29	0.1119	−25.2381	−25.3064
0.10	432.03	0.1167	−32.7698	−32.9619
0.20	357.52	0.1484	−29.9510	−30.2127
1.00	325.32	0.1981	−44.8659	−46.3816
2.00	300.25	0.2567	−49.5137	−51.8243
5.00	249.44	0.4445	−43.3256	−48.8986
10.00	215.01	0.6281	−28.4257	−38.3975

a: CPU time (in minutes) cost by a single run of ACMC on an Intel Core(TM)2 DUO computer.
b: The minimum energy sampled by a single run of ACMC.
c: The minimum energy sampled by a single run of the post Metropolis moves.

5. Conclusion

We have shown in this paper that the ACMC algorithm can be effectively applied to the prediction of the protein folding on an off-lattice BLN model. Numerical results favor to ACMC in finding the ground states, and thus show that ACMC is a very promising algorithm for a general optimization task.

A further work of interest is how to balance the energy. The ACMC algorithm is very effective for all proteins, but it is unstable from the standpoint of the root mean square deviation between the structure of folded target protein and the native structure. The use of the ACMC algorithm will be further explored to reduce its instability.

References

Gelman, A., Roberts, G. O. and Gilks, W. R. (1996). Efficient Metropolis jumping rules, *In bayesian statistics 5*, edited by J.M. Bernardo, J.O. Berger, A.P. Dawid, and A.F.M. Smith, 599–607. Oxford University Press.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Professional, Michigan.

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.

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

Kim, S. Y., Lee, S. J. and Lee, J. (2003). Conformational space annealing and an off-lattice frustrated model protein. *Journal of Chemical Physics*, **119**, 10274–10279.

Kim, S. Y., Lee, S. B. and Lee, J. (2005). Structure optimization by conformational space annealing in an off-lattice protein model, *Physical Review E*, **72**, 011916-1–011916-6.

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

Li, Z. and Scheraga, H. (1987). Monte Carlo-minimization approach to the multiple-minima problem in protein folding, *Proceedings of the National Academy of Sciences of the United States of America*, **84**, 6611–6615.

Liang, F. (2004). Annealing contour Monte Carlo algorithm for structure optimization in an off-lattice protein model, *Journal of Chemical Physics*, **120**, 6756–6763.

Liang, F. (2005). A generalized Wang-Landau algorithm for Monte Carlo computation, *Journal of the American Statistical Association*, **100**, 1311–1327.

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**, 1087–1092.

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.

Zhang, X., Lin, X., Wan, C. and Li, T. (2007). Genetic-annealing algorithm for 3D off-lattice protein folding model, *Lecture Notes in Computer Science*, **4819**, 186–193.