Monte Carlo methods designed for parallel computation

Sheldon B. Opps and Jeremy Schofield
Chemical Physics Theory Group, Department of Chemistry,
University of Toronto, Toronto, Ontario, Canada M5S 3H6
(September 28, 2001)

In this paper various extensions of the parallel-tempering algorithm are developed whose inherent structure makes them easy to implement on a Beowulf cluster. The generic feature of these methods is that individual Monte Carlo chains, which are run on a separate nodes, are coupled together to form a composite chain. Communication between nodes is accomplished with routines from the Message-Passing Interface (MPI) libraries. By monitoring this communication, the methods can be tuned for a given architecture. The algorithms are tested on a model *in vacuo* polypeptide system for which it is demonstrated that the competing factors of chain-coupling frequency and messaging overhead can be adjusted in order to achieve optimal performance. Optimization issues are discussed in detail for calculations performed on a Beowulf cluster consisting of 30 600-MHz alpha processors linked over a switched network.

I. INTRODUCTION

The use of supercomputers to perform highly intensive numerical calculations has allowed researchers to probe into areas of science previously not accessible by other means. Such tools have provided answers to some fundamental questions and have lead to significant advancement in areas such as complex fluids [1] and meteorological modeling [2]. Recently, there has been a trend towards building smaller-scale "Beowulf" clusters [3], consisting of a group of processors (nodes) which are linked together to form a local network. Through the use of high-speed switches, it is possible for the nodes to communicate between each other at fast rates and thus provide the opportunity to perform larger-scale calculations at a fraction of the financial cost of that of mainframe computers.

Although one may use each node to execute a separate calculation, for example to improve the statistics of a Monte Carlo simulation, one inherent benefit of these clusters is the ability to utilize all of the nodes in parallel to perform a single calculation. Depending on the nature of the computer code, and optimization issues, such a parallel architecture can dramatically increase the speed of a given numerical calculation. This parallel design can be used in an even more effective manner when studying systems that are characterized by rough energy landscapes, such as spin glasses [4] or biological molecules [5]). When using standard Markov chain Monte Carlo procedures to study such complex systems, there is a propensity for the system to become configurationally trapped in isolated modes [6] and the sampling mobility is hindered.

A number of methods have been devised to deal with such "quasi-ergodic" sampling. These techniques rely on broadening the sampling distribution using umbrella-sampling techniques [7] including multi-canonical methods [8]. Another sampling scheme, known as parallel tempering [9], which we term "simple swapping", has been proposed which is designed to increase the mobility of the Markov chain by sampling on an extended state space. Recently, we have proposed a new method which incorporates an annealing procedure intended to guide the extended system through the auxiliary parameter space in a directed fashion [10]. This method, referred to as "annealed swapping", is based upon both annealed-importance sampling [11] and multiple Markov chain methods [12].

In both the simple and annealed swapping procedures, individual Markov chains with different limiting distributions are coupled together to form a composite chain. If these individual chains are generated on *separate* processors or nodes of a parallel computer, then the coupling procedure relies on communication between the nodes and, hence, the method is well suited for parallel computation. The main focus of this paper is to demonstrate how such methods can be used on a Beowulf cluster in order to improve both the sampling mobility and computational efficiency of the algorithms. By adjusting a number of tunable parameters, such as the number of nodes and the frequency of communication between nodes, one can optimize the efficiency of the computation for a given architecture. In this paper, we test both the simple and annealed swapping algorithms on a model system composed of a single, 10 residue polyglycine chain *in-vacuo*, where these optimization issues are discussed in greater length. The simulations are performed on a Beowulf cluster consisting of 30 600-MHz alpha processors linked over a switched network.

II. ALGORITHMS

A. Parallel tempering

The basic idea of the parallel tempering [12,13] or simple-swapping algorithm is that the mobility of isolated Monte Carlo chains with different limit distributions can be enhanced by coupling them together to form a composite Markov chain whose limit distribution is the product distribution of the separate chains. The method works particularly well for chains in which the convergence is slow, since the other chains in the composite chain which have greater mobility act as a randomizing heat bath which destroys correlation.

In this algorithm standard Monte Carlo updates are performed independently on a set of chains which span a range of parameters (such as temperature). After a certain number of updates, two chains are selected (either randomly or sequentially) and an attempt is made to "swap" the current configurations on the two chains. Although it is possible to select non-adjacent chains, in practice it is often best to select only adjacent chains to test for swapping since the probability of acceptance diminishes as the overlap of the chains limiting distributions decreases.

One of the adjustable parameters in this method is the number of chains. It might be surmised that a large number of chains is optimal, since this would increase the probability that a swap would be accepted. However, in practice, this is not found to be true since as the swap acceptance ratios approach one a given configuration evolves in the parameter space in a random walk. For systems, such as protein-folding studies, which require a large range of temperatures due to the depth of the attractive basins in the potential energy surface, the diffusive nature of the algorithm may lead to slow exploration of the entire parameter space. It is therefore worthwhile to consider variants of the simple-swapping algorithm in which the exploration of the parameter space is not stochastic.

B. Annealed swapping

One means of addressing diffusive exploration of the auxiliary parameter space is to develop a Monte Carlo updating scheme which enables configurations on non-adjacent chains to be swapped with non-negligible probability. This may be achieved by constructing an updating scheme utilizing an *annealing schedule* which includes both a quenching and heating procedure. The simulated annealing procedure [11] has been used to help dislodge systems trapped in low-energy configurations (such as for glassy systems). However, one of the drawbacks of this procedure is that statistical information about the relative statistical importance of each structure is not accessible. Recently, a Monte Carlo algorithm [14] has been developed, "annealed importance sampling", which uses a procedure similar to simulated annealing but calculates statistical weights for the annealing process so that an importance sampler may be defined.

We have incorporated such a procedure within the context of the parallel tempering algorithm [10]. Such an annealing process, which we refer to as "annealed swapping", is devised as follows. Consider two *principal* chains at inverse temperatures β_1 and β_2, $\beta_1 > \beta_2$, with corresponding limiting distributions P_1 and P_2. The two chains are updated in parallel using standard Monte Carlo. After a certain number of Monte Carlo (MC) updates or "steps", the current configurations $\hat{x}_1$ and $\check{x}_2$ (where $\hat{}$ denotes heating and $\check{}$ refers to cooling) on β_1 and β_2 are used as starting points for heating and cooling processes, respectively, along a set of n *intermediate* paths which span between the two principal chains. This procedure generates a set of intermediate configurations with associated statistical weights $\hat{w}$ and $\check{w}$. The method is schematically depicted in Fig. 1.

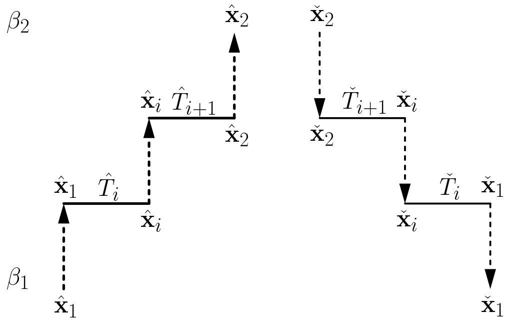

FIG. 1. A schematic of the annealing path $\{\hat{x}_1, \hat{x}_2, \check{x}_2, \check{x}_1\}$. The solid lines indicate Monte Carlo updates according to the transition probabilities $\hat{T}_i$ and $\check{T}_i$.

After this process is completed, one tests to see if the configurations $\hat{x}_2$ and $\check{x}_1$ will be accepted as the new states for chains 1 and 2, respectively. The probability for accepting such a move is given as $P_A = \min(1, \hat{w} \times \check{w})$. More Details about this algorithm can be found in an earlier paper [10].

III. PARALLEL COMPUTATION

One of the nice features of both the simple and annealed-swapping methods is that they can be implemented on a parallel computer. Each of the principal Markov chains can be run on separate nodes and regular MC updating is performed for a given number of steps. When the updating has been completed for all of the chains (nodes), an attempt is made to execute either a simple or annealed swap of configurations between adjacent chains. Such swapping between nodes requires the use of communication tools, such as the MPI message-passing interface libraries. for communication between the nodes. Typically, one "master" node is used as a conductor to coordinate the communication between the nodes. Once a swapping attempt has been made, standard updating resumes along the individual chains and the nodes operate as if they were independent.

Key issues to consider when designing computer code to be used in such a parallel fashion are load balancing and communication overhead: the cost of communicating between the nodes. Some of the relevant factors to consider are "how much data are being transferred between nodes" and "how long it takes to perform this transfer". Although the basic swapping algorithm for both of the above methods involves swapping configurations, it is much more efficient to swap the *parameters* which distinguish the different chains. In the case of a simple swap between two chains at inverse temperatures β_1 and β_2, this translates into swapping the *inverse temperatures* rather than the configurations between the chains. The net result is the same in that one obtains a composite chain which spans the extended-state space.

Not only is the time for data transfer between nodes an important factor, but also the frequency at which the nodes communicate is critical. Such a variable is controlled by the number of updating steps along a given principal chain: the shorter the updating, the more frequently swaps will be performed between nodes. Thus, although swapping may lead to more efficient sampling of phase space, frequent swapping may incur a significant computational cost. One must therefore carefully tune the algorithm (either simple or annealed swapping) in order to balance these opposing factors.

There are a number of other adjustable parameters which affect the performance of the sampling methods. As noted above, increasing the number of principal chains may increase the swap acceptance probabilities, but this results in greater communication between nodes and can be costly. Similarly, the number of intermediate updating steps in the annealed-swapping algorithm can be varied to allow for improved sampling mobility. In the following section, we investigate these optimization issues on a 10 residue, *in-vacuo* polyglycine molecule.

IV. RESULTS

A. Peptide Model

We have performed Monte Carlo studies on gly10, a peptide composed of 10 glycine residues, in which the CH_2 methylene groups were treated as single monomer units under the united atom scheme. Simulations were performed with the annealed-swapping method and, for comparison, the simple-swapping algorithm over a temperature range of $400-1000\,\mathrm{K}$. The total number of MC steps for the simulations was 5×10^7 with a burn-in length of $\sim 1\times10^3$ steps. It is convenient to re-express the total number of updates N_{up} for the simulation in terms of the the number of steps per swap attempt which, for N principal chains, n intermediate chains, and m updates along a chain, consists of $2(N-1)mn$ steps/swap. To determine the optimal combination of the above parameters, the "trajectory" of a component of the composite chain was followed as it cycled from lowest to highest temperatures as a means of monitoring the efficiency of the algorithm. A single cycle was measured as the completed path, beginning at the lowest temperature, traversing to the high temperature chain, and finally returning to the lowest temperature.

For the annealed-swapping simulations, it was determined that the number of cycles per CPU second, from here on referred to as the "mixing rate", was optimized for a system of $N=3$ principal chains or "threads"(with $T = 400, 600, \mathrm{and}\, 1000\,\mathrm{K}$) with $n = 80$ intermediate threads and $m = 156$ updating steps along each thread, corresponding to 1000 swaps with 5×10^4 total updates per swap . The simple-swapping runs utilized 6 principal chains (with $T = 400, 470, 550, 650, 800, 1000\,\mathrm{K}$) with 50 updates per swap along a chain, giving 166,666 swaps with 300 steps per swap. For both the annealed and simple-swapping algorithms, each of the principal N chains were run on N independent nodes on the Beowulf cluster, where one of these nodes was chosen as the master communicating node. After the m updates were completed, all of the nodes were interrupted by the master node, M, and a swap attempt was made (either simple or annealed) between adjacent chains on different nodes. Once a swapping cycle was completed, the master M would pass the relevant data to all of the nodes and regular updating would resume independently on each of the N nodes.

One of the important thermodynamic quantities to consider for such studies is the radius of gyration, R_g, which gives a measure of the "size" of the peptide. In Fig. 2, data are presented for R_g for a particular component of the composite chain utilizing the annealed-swapping procedure.

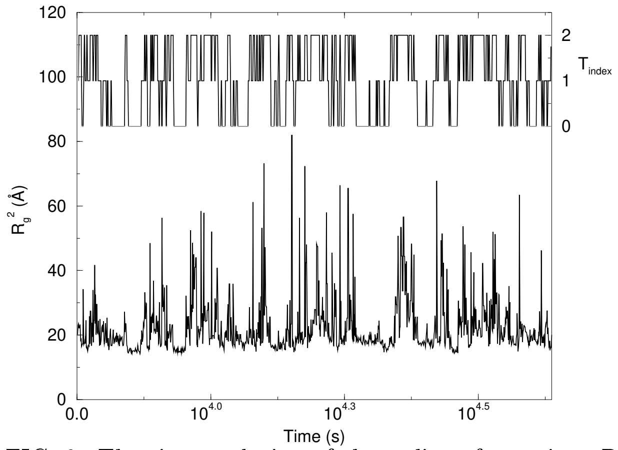

FIG. 2. The time-evolution of the radius of gyration, R_g, for a component of the composite chain for a simulation of gly10 employing the annealed sampling method.

Comparing the temperature index T_{index} to R_g, one observes that at low temperatures the peptide adopts a compact structure and at higher temperatures the chain becomes more extended. This can also be seen clearly in Fig. 3, where "snapshots" of the peptide at different times in the simulation are illustrated.

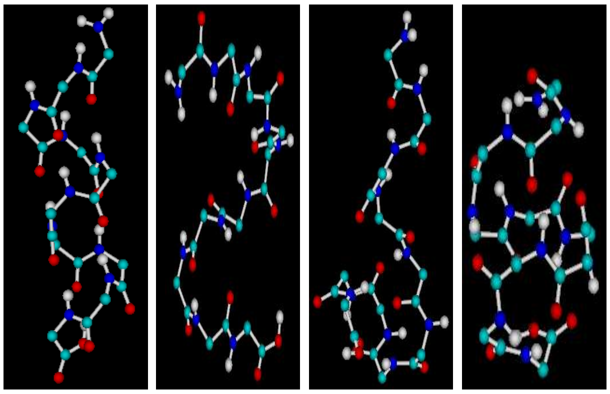

FIG. 3. Snapshots of gly10 along a component of the composite chain at different times in the simulation.

It is evident that this component of the composite chain samples the full temperature range and becomes thoroughly mixed.

To compare the annealed and simple-swapping methods, we consider the weights associated with swapping between two adjacent chains. The histograms of these weights are plotted in Fig. 4.

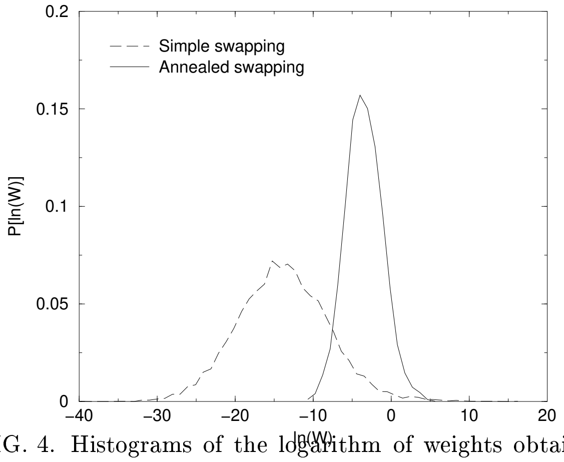

FIG. 4. Histograms of the logarithm of weights obtained from simulations of gly10 for both the simple and annealed-swapping methods.

There is a significant proportion of the annealed-swapping distribution which lies greater than zero, resulting in good overlap between the distributions of the chains. The increased overlap of the distributions of the chains results in an enhanced swapping probability. However, despite better overlap between adjacent chains for the annealed-swapping procedure, the mixing rates are an order of magnitude smaller than for the simple-swapping algorithm.

Hence, despite better overlap between adjacent chains, the annealed-swapping technique did not perform as well as the simple-swapping algorithm. A possible explanation for these discrepancies is that, although annealed swapping results in greater overlap between chains, there is a greater CPU cost for swap attempts. For annealed swapping, much time is expended performing standard updating on the intermediate chains. If the move is rejected, then much work is wasted since the weights generated on the intermediate threads are not utilized. Thus, in this scheme, failed attempts are quite costly. In contrast, for simple swapping, failed swap attempts are computationally cheaper than accepted attempts (which require a shuffling of temperatures and chain labels), since no updating is required. One can take advantage of this mismatch by attempting more frequent swap moves which, although resulting in lower acceptance ratios, allows for greater mixing of the chains per CPU second. In particular, a surprising observation was that the *greatest* mixing rates were obtained by performing *many* swap attempts with short updating. Although there is an increased overhead due to such frequent communication between nodes, this is more than compensated for by the improved performance of the algorithm. We anticipate that this is a general characteristic of the method and is not system dependent. It should be mentioned that such communication overhead is negligible in light of the dramatic improvement in execution time for simulations performed on a multi-node cluster compared to those on a single processor.

Note that to reduce the number of failed swap attempts for annealed swapping, since failed attempts are costly, and improve the acceptance probabilities, one can increase the number n of intermediate threads and adjust the number m of updates. However, this also comes at a cost due to the increased time for regular Metropolis updating. As mentioned above, we have performed detailed analysis and ascertained that there is an optimal number of intermediate chains, beyond which one finds diminishing returns. We have determined that these findings are not altered by modifications of the standard updating scheme along an individual chain. It is thus evident that the regular Monte Carlo updating is the key bottleneck in these computations and *not* swapping frequency or inter-node communication.

These are encouraging results which demonstrate that such extended-state Monte Carlo methods are well suited for use on a parallel computer/architecture.

V. CONCLUSIONS

In this article, we have tested two extended-state Monte Carlo methods, simple and annealed swapping, on a model *in vacuo* polypeptide system. The inherent structure of these algorithms permits them to be used for parallel computation on a Beowulf cluster. There are a number of adjustable parameters for these methods which can be optimized to achieve the best performance for a given cluster architecture. Some of these parameters are the number of principal chains, the number of standard updating steps and swapping frequency, and, for annealed swapping, the number of intermediate chains. In this work, extensive studies have been done to ascertain the most optimal set of parameters.

An interesting result of these investigations was that swap attempts should be executed extremely frequently in the simple-swapping approach. This result seems surprising in light of the fact that rapid swaps require short intervals of local Metropolis updates which induces strong correlations in the swap dynamics of the chains. It appears that, although increasing the length of the local updates tends to destroy correlations, the rate at which correlations are lost is not sufficient to compensate for the reduced swap time. These results also indicate that the performance of the algorithm is insensitive to the increased communication overhead associated with rapid swapping between nodes. This is an important finding which indicates that the method is not only inherently well suited for parallel computation but that significant optimization can be achieved for a given implementation.

It has also been demonstrated that, although the annealed-swapping procedure increases the swap probability by effectively improving the overlap of the energy distributions of adjacent chains, the mixing rates were greater for simple swapping. With annealed swapping, a large percentage of the CPU time is dedicated to performing simple Metropolis updates on the intermediate threads. Hence, although the number of failed swap attempts is reduced by the improved overlap between adjacent chains, much time is consumed on average to obtain an accepted swap. For a fixed number N_{up} of total MC steps for the simulation, the increased amount of standard updates results in a much reduced number of swap attempts in comparison with the simple-swap procedure. If communication between nodes was a critical factor governing the efficiency of the methods, then one would expect that the reduced amount of swapping, in conjunction with optimal chain distribution overlap, would result in *improved* performance of the method. The fact that such improvement in efficiency is not observed again supports the conclusion that these methods involve minimal communication overhead.

Although there was no significant improvement found in the performance of the annealed-swapping method over that of simple swapping, it is possible that for a more complex system, where a larger parameter space is required in order to guarantee good sampling mobility, that the simple-swapping method would become less efficient as the parameter-space exploration becomes increasingly more diffusive. In such a case, it may be found that annealed swapping would be a superior method. Further studies to examine these issues are currently being pursued.

ACKNOWLEDGMENTS

The authors would like to thank Timothy Guimond for assistance in coding the algorithms. This work was supported by a grant from the National Sciences and Engineering Research Council of Canada.

[1] J.-.C. Desplat, I. Pagonabarraga, and P. Bladon, Computer Physics Communications, **134**, 273, (2001).

[2] A. Lewis, D. Abamson, and R. Simpson, in *SC97 Proceedings*, 1997.

[3] D. Ridge, D. Becker, P. Merkey, and T. Sterling, in *Proceedings, IEEE Aerospace*, 1997.; C Reschke, T. Sterling, D. Ridge, D. Savarese, D Becker, and P. Merkey, in *Proceedings, High Performance and Distributed Computing*, 1996.

[4] W. Kerler and P. Rehberg, Phys. Rev. E **50**, 4220, (1994); B. A. Berg and T. Celik, Phys. Rev. Lett. **69**, 2292 (1992).

[5] U. H. E. Hansmann and Y. Okamoto, J. Comp. Chem. **14**, 1333 (1993); M.-H. Hao and H. A. Scheraga, J. Phys. Chem. **98**, 4940 (1994).

[6] J. P. Valleau and S. G. Whittington, in *Statistical Mechanics*, edited by B. J. Berne (Plenum, New York, 1977), Ch. 4, pp. 114-119.

[7] G. M. Torrie and J. P. Valleau, J. Comput. Phys. **23**, 187 (1977); J. P. Valleau, J. Chem. Phys. **99**, 4718 (1993).

[8] B. A. Berg, Int. J. Mod. Phys. C **3**, 1083 (1992); U. H. E. Hansmann, Chem. Phys. Lett. **281**, 140 (1997).

[9] E. Marinari and G. Parisi, Europhys. Lett. **19**, 451 (1992); A.P. Lyubartsev, A. A. Martinovski, S. V. Shevkunov, P. N. Vorontsov-Velyminov, J. Chem. Phys. **96**, 1776 (1992).

[10] S. B. Opps and J. Schofield, Phys. Rev. E. **63**, 056701 (2001).

[11] S. Kirkpatrick, C. D. Gelatt, and P. M. Vecchi, Science **220**, 671 (1983).

[12] C. J. Geyer in *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, 156, Interface Foundation.

[13] K. Hukushima and K. Nemoto, J. Phys. Soc. (Jap.) **65**, 1604 (1996).

[14] R. M. Neal, *Technical Report No. 9805*, Dept. of Statistics, Univ. of Toronto, 1998.