Proceedings of the 51^{st} Hawaii International Conference on System Sciences | 2018

Stochastic Speculative Computation Method and its Application to Monte Carlo Molecular Simulation

Yasuki Iizuka*, Akira Hamada, Yosuke Suzuki
Graduate School of Science, Tokai University
Hiratsuka, Japan
* iizuka@tokai-u.jp

Abstract

Monte Carlo (MC) molecular simulation has significant computational complexity, and parallel processing is considered effective for computation of problems with large complexity. In recent years, multicore or many-core processors have gained significant attention as they enable computation with a large degree of parallelism on desktop computers. However, in conventional parallel processing, processes must be synchronized frequently; thus, parallel computing is not necessarily efficient. In this study, we evaluate the effect of applying MultiStart-based speculative parallel computation to MC simulations. Using probability theory, we performed theoretical verification to determine if speculative computation is more effective than conventional parallel computation methods. The parameters obtained from the theoretical calculations were observed in experiments wherein the speculative method was applied to an MC molecular simulation. In this paper, we report the results of the theoretical verification and experiments, and we show that speculative computation can accelerate MC molecular simulations.

Keywords: speculative computing, parallel processing, molecular simulation

1. Introduction

Recently, computer-based molecular simulations, e.g., molecular dynamics (MD) and Monte Carlo (MC) simulations, that calculate the characteristics of a molecule have attracted significant attention. Both MD and MC simulations have considerably large computational complexity. It is not uncommon for these calculations to take weeks or months. To make molecular simulation easier to use, it is essential to speed up the calculation. The theoretical basis of the MC simulation is an ergodic Markov chain; thus, MC simulations are similar to the simulated annealing (SA) algorithm, which is used to solve combinatorial optimization problems.

Parallel processing can effectively accelerate the computation of problems with large complexity, e.g., combinatorial optimization problems. However, even if m parallel processes (or threads) are executed, the execution time will not be reduced to $1/m$ because (i) only some of the processes can be parallelized, (ii) the process generates overhead, and (iii) some processes must be synchronized. Therefore, the capacity factor of parallel computing resources will not be 100%.

Parallelization can also be performed using speculative computation. Speculative computation is to pre-execute calculations that may not use calculation results in the future. For example, Multilisp[1] or MultiStart [2] executes speculative computation simultaneously (or in pseudo parallel).

The purpose of this research is to develop a computation method that accelerates molecular simulations. Therefore, we examine the performance of a MultiStart-based speculative computation method when it is applied to a real problem. In this study, the speculative method uses a stochastic algorithm, such as the SA algorithm or Tabu search, as the base program. We execute multiple processes that use a common base stochastic program in parallel with a different random number seed and select the best solution.

We show the results of advanced theoretical estimations using probability theory. In addition, we perform an experiment to computationally solve an actual problem using the speculative method. The results demonstrate that speculative computation is more effective than conventional parallel computing for some problems.

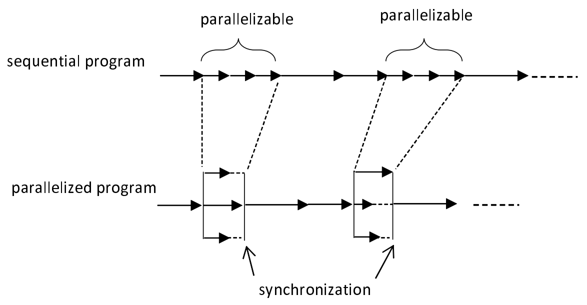

Figure 1. Example of a conventional parallelized program

2. Parallel Processing

Many parallel programs are developed from sequential programs to reduce computation time, i.e., some parts of sequential programs can be parallelized. Figure 1 shows an example of a parallelized program. Such programs incur process generation and synchronization costs. When synchronization is required, completed processes must wait for running processes. Processing cannot continue during this waiting state; thus, computational resources are not used efficiently. Increasing the number of processes does not always increase processing speed. For example, if a part of a program is executed using 10 processes, the execution time does not necessarily increase by a factor of 10.

The ratio of the part of a program that can be parallelized relative to the entire program is assumed to be α $(0 \leq \alpha \leq 1)$. When this part is parallelized by m processes (or threads), the computation time is expressed as follows:

$$\alpha/m + (1 - \alpha) + \beta, \tag{1}$$

where β is the process generation time and synchronization wait time. When α is sufficiently close to 1, the computation time decreases to $1/m + \beta$. In many cases, α does not attain such an ideal value.

Studies reporting the process to parallelize SA [3], [4] and Tabu search [5] were published more than 20 years ago. However, in these studies, synchronization was required in all iterations.

In speculative parallel computation, a MultiStart [2] procedure executes several computations speculatively and simultaneously, including computations that may not be used. In theoretical analyses using the state transition matrix of the Markov process, it has been suggested that MultiStart can realize superlinearity [6], [7]. However, in these analyses, superlinearity was evaluated using theoretical comparisons based on a simple random search. To the best of our knowledge, experiments targeting practical problems other than benchmarks have not been performed. Note that it is difficult to realize superlinearity by comparing efficient algorithms using various heuristics.

Therefore, various speculative computation methods have been proposed [8], [9], [10]. The line-up competition algorithm (LCA) [11] is a speculative parallel computation algorithm that attempts to obtain the best result after executing many programs. LCA compares solutions among process groups, which are referred to as a family, at every iteration that yields an improvement and modifies the program parameters that are dynamically based on a comparison of the results. In LCA, this comparison is performed after each iteration; consequently, frequent synchronization is required. The synchronization cost of the speculative computation method used in our experiment is nearly zero; moreover, it can parallelize the existing algorithm without modification. In addition, wait time is not required; thus, nearly 100% of the computational resources can be used.

The algorithm portfolio method has also been proposed [12] as a type of promising speculative computation method for difficult problems. In the algorithm portfolio, some (one or more) algorithms are executed simultaneously; then, the algorithm with the best performance is selected. In our study, only one algorithm was executed in parallel; this may be a precise analysis of a special example of an algorithm portfolio. However, assuming only one algorithm, theoretical calculation of the expected value becomes possible. Furthermore, in molecular simulations, multiple algorithms cannot be used and parallel execution of a homogeneous algorithm is required to maintain the validity of the simulation.

In previous studies, the effect of speculative computation was theoretically analyzed using the state-transition matrix eigenvalues of the Markov process [6], [7]. However, in an actual problem, it is difficult to obtain an accurate state-transition matrix. Therefore, only the lower or upper bound can be considered. In this study, we attempt to analyze the effect of the speculative method as a simple probability problem without using a state-transition matrix and derive parameters that can be observed experimentally; i.e., we attempt to analyze the application of this method to a practical problem.

3. Effect of Speculative Computation

3.1. Algorithm

Various successful metaheuristics such as SA, Tabu search, and genetic algorithms (GA) have been proposed for combinatorial optimization problems. Such metaheuristics use the stochastic iterative improvement algorithm as the base. If these algorithms run for a very long time, the obtained solution will approach the global optimal solution [13].

In this paper, we make the following assumption. Here, let t denote the iteration step, ϵ denote the optimal solution, and a and b denote constants. The expected value μ of the solution obtained by the stochastic iterative improvement algorithm, which solves combinatorial optimization problems, is expressed as a function of the iteration step t in the following equation.

$$\mu = a \cdot t^{-b} + \epsilon. \tag{2}$$

This assumption will be verified on each experiment in Section 4 and 5.

In addition, we assume abundant parallel computing resources, such as multicore CPUs, many-core CPUs, or a computer cluster.

In this study, we attempt to analyze the effect of the speculative computation method as a probability problem.

When the results of a trial S conform to a specific probability distribution, if the trial is repeated and the minimum value of the repeated trials is selected as trial M, the latter conforms to a probability distribution that differs from trial S. For example, if the scores are determined by a dice roll, the probability distribution becomes 1/6 each of $X = \{1, 2, 3, 4, 5, 6\}$ with an expected value of 3.5 for μ_s. If m dice are thrown simultaneously and the scores resulting from the minimum value are considered to be trial M, the probability distribution will be nonuniform; moreover, as the value of m increases, the expected value μ_m will become closer to 1.

This is the principle of speculative computation. In other words, if a stochastic algorithm is simultaneously executed m-parallel and the minimum value is selected as the result, it may be possible to obtain better computational performance.

In this study, a stochastic algorithm is used as the base and the following extremely simple speculative method is adopted.

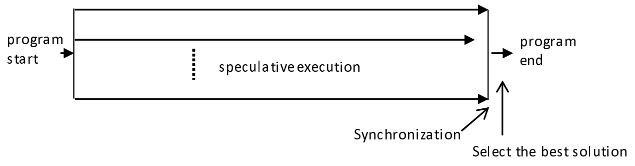

Figure 2. An image of speculative computation

Step 1. A stochastic algorithm is executed independently using multiple parallel processes (or threads), where a different initial value or different random number seed is used for each process.

Step 2. All parallel executions are terminated after a fixed time or fixed number of iterations. Then, the best solution is selected from the obtained solutions.

The solution obtained by this method varies stochastically with parameter m. A conceptual diagram of this method is shown in Figure 2. This method represents a simple Multi-Start, which we refer to as Stochastic Speculative Computation (SSpeC). SSpeC has the following features.

1) Process generation is performed once, and the process generation cost is low.
2) Synchronization is performed once, and the wait time is short.
3) The program utilizes nearly 100% of the available computational resources.
4) The program can be parallelized without modifying the original program.

3.2. Effect of improving a solution

It is assumed that the solution to a stochastic algorithm $\mathcal{S}$ follows the probability distribution of the distribution function $F_s(y)$ and the probability density function $f_s(y)$, where the minimum solution (or the optimal solution) is denoted as ϵ. $\mathcal{S}$ is executed in an m-parallel manner speculatively, and the minimum solution is selected from the obtained solutions. Note that m-parallel execution is assumed to be independent trial. In this case, the probability that solution y can be obtained via m-parallel execution follows the minimum value distribution of the original probability distribution according to the extreme value theory. The probability distribution function $F_m(y)$ and probability density function $f_m(y)$ are expressed as follows:

$$F_m(y) = 1 - (1 - F_s(y))^m, \tag{3}$$

$$f_m(y) = m(1 - F_s(y))^{(m-1)} f_s(y). \tag{4}$$

Here, $F_s(y)$ is sufficiently small, i.e., it is close to the minimum value of the original probability distribution. Therefore, it can be approximated as $ln(1-F_s(y)) \approx -F_s(y)$. Using this approximation, we can derive the following as an approximate asymptotic distribution of the minimum value distribution:

$$\begin{aligned} F_m(y) &= 1-(1-F_s(y))^m \\ &= 1-e^{ln(1-F_s(y))^m} \\ &\approx 1-e^{-mF_s(y)}. \end{aligned} \tag{5}$$

The shape of the foot portion near the minimum value of the original distribution characterizes the minimum value distribution (Eq. (5)). Therefore, generally, by analyzing the extreme value distribution of the minimum value, a type-III asymptotic minimum value distribution, which assumes the existence of a lower limit in the original probability distribution, is used [14]. For a type-III asymptotic minimum value distribution, the following simple model, which is a distribution of the lower limit value ϵ (the optimal value) and the upper limit value τ, can be considered. Here, the rise around the minimum value is a power form. This model is expressed as follows.

$$\begin{aligned} F_s(x) &= \left(\frac{x-\epsilon}{\tau-\epsilon}\right)^k \\ &= \left(\frac{x-\tau}{\delta}\right)^k, \quad (\epsilon \le x \le \tau). \end{aligned} \tag{6}$$

$$f_s(x) = \frac{k}{\delta}\left(\frac{x-\epsilon}{\delta}\right)^{k-1}, \quad (\epsilon \le x \le \tau). \tag{7}$$

Here $\delta = \tau - \epsilon$. Substituting Eq. (6) into the Eq. (5) yields the following:

$$\begin{aligned} F_m(y) &= 1-e^{-mF_s(y)} \\ &= 1-e^{-m((y-\epsilon)/\delta)^k}. \end{aligned} \tag{8}$$

Here, let $\theta = \delta/m^{1/k}$. Then, the probability distribution function is expressed as follows:

$$F_m(y) = 1-e^{-((y-\epsilon)/\theta)^k} \tag{9}$$

$$f_m(y) = \frac{k}{\theta}\left(\frac{y-\epsilon}{\theta}\right)^{k-1} e^{-((y-\epsilon)/\theta)^k}. \tag{10}$$

This is a Weibull distribution. The average value of this probability distribution can be obtained as follows [15]:

$$E(Y_m) = \int_\epsilon^\infty y\,\frac{k}{\theta}\left(\frac{y-\epsilon}{\theta}\right)^{k-1} e^{-((y-\epsilon)/\theta)^k} dy. \tag{11}$$

The integration range is originally $\epsilon \sim \tau$; however, computationally, it is $\epsilon \sim \infty$. Here, we consider the following variable transformation.

$$z = \left(\frac{y-\epsilon}{\theta}\right)^k \tag{12}$$

$$y = \epsilon + \theta z^{1/k} \tag{13}$$

$$dz = \left(\frac{k}{\theta}\right)\left(\frac{y-\epsilon}{\theta}\right)^{k-1} dy \tag{14}$$

Then, the integration range $y = \epsilon \sim \infty$ changes to $z = 0 \sim \infty$. Therefore, the average is expressed as follows.

$$\begin{aligned} E(Y_m) &= \int_0^\infty \left(\epsilon + \theta z^{1-1/k}\right)\frac{k}{\theta}e^{-z}\left(\frac{\theta}{k}\right)z^{-1+1/k}dz \\ &= \int_0^\infty \left(\epsilon + \theta z^{1/k}\right)e^{-z}dz \\ &= \epsilon + \theta\int_0^\infty z^{1/k}e^{-z}dz \\ &= \epsilon + \theta\int_0^\infty z^{(1+1/k)-1}e^{-z}dz \\ &= \epsilon + \theta\,\Gamma(1+\frac{1}{k}). \end{aligned} \tag{15}$$

Here, $\Gamma(\ )$ is a gamma function and δ and k are parameters that depend on the shape of the original probability distribution. In this paper, we want to express the average value as a function $\mu_m(m)$ of the number of processes m. By returning θ to its original form, we obtain the following.

$$\mu_m(m) = E(Y_m) = \epsilon + \frac{\delta}{m^{1/k}}\Gamma(1+\frac{1}{k}). \tag{16}$$

If the expected values of the probability distribution $\mathcal{S}$ is μ_s, since $\mu_m(1) = \mu_s$, we simplify (16) as follows.

$$\begin{aligned} \mu_s &= \epsilon + \frac{\delta}{1^{(1/k)}}\Gamma(1+\frac{1}{k}) \\ \mu_s - \epsilon &= \delta\Gamma(1+\frac{1}{k}). \end{aligned} \tag{17}$$

We then substitute this into Eq. (16), and substitute $1/k$ for h ($h = 1/k$) to obtain the following:

$$\mu_m(m) = \epsilon + m^{-h}(\mu_s - \epsilon). \tag{18}$$

Similarly, the distribution $\sigma_m^2(m)$ is expressed as follows:

$$\sigma_m^2(m) = \left(\frac{\delta}{m^{1/k}}\right)^2\left(\Gamma\left(1+\frac{2}{k}\right) - \Gamma\left(1+\frac{1}{k}\right)^2\right). \tag{19}$$

In addition, $\sigma_m^2(1) = \sigma_s^2$ can be used to describe the distribution as follows:

$$\sigma_m^2(m) = m^{-2h}\sigma_s^2. \tag{20}$$

Here, h is an index of the speculative computation effects, and a greater value of h results in greater speculative

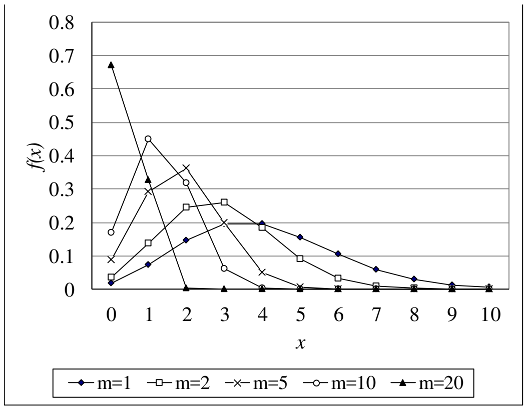

Figure 3. Conceptual graph of a probability distribution shift by speculative computation

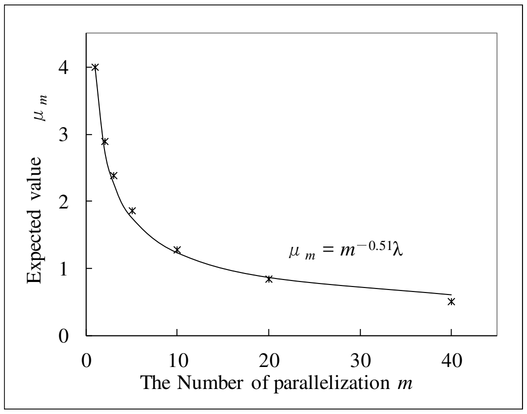

Figure 4. The relationship between the number of parallelization (= the number of processes) m and the expected value μ_m when F_s is the Poisson distribution

computation effects. From Eq. (18), the effects of speculative computation can be expressed as m^{-h} (the power function of m). The index of the effects of the speculative computation h is the inverse of k; therefore, the distribution will follow the shape of the original probability distribution. When the parallel number is $m \to \infty$, the expected value μ_m approaches ϵ asymptotically.

A conceptual graph of how the probability distribution of speculative computation changes is shown in Figure 3. Here, a Poisson distribution is assumed to be the original probability distribution for the parameter $\lambda = 4$. In addition, there are the results of the numerical calculation of the probability distribution when parallelizing with $m = 1, 2, 5, 10, and$ 20 ($m = 1$ is the original probability distribution). When m is increased, the mode of the probability distribution slightly shifts to the left and the variance becomes small. Movement of the mode occurs in the same way even for original non-Poisson probability distributions.

We calculated the expected value μ_m of the probability distribution with increasing m. Figure 4 shows the result of the calculation when the trial of Figure 3 is parallelized. The approximate curve $\mu_m = m^{-0.51}\lambda$ has been added to Figure 4. Here, h in Eq.(18) can be obtained from this curve. Such reduction of the expected value is not limited to the case where the original probability distribution is Poisson. The results of our numerical analysis indicate that when the original probability distribution is a bell curve (similar to a normal distribution), h is in the range $0 < h < 1$. When the original probability distribution has a long tail (similar to geometric or exponential distributions), $h > 1$ may be observed. If $h > 1$, the effect of speculative computation is superlinear. In other words, h becomes large when the distribution of the original probability distributions is large. If the original probability distribution $F_s(x)$ is distributed uniformly, the distribution $F_m(x)$ of the speculative computation becomes a beta distribution and $h = 1$.

Generally, when a problem is difficult, the distribution of the solution of a stochastic algorithm becomes large, i.e., the effect of speculative computation is significant for difficult problems. Further, it is important that h, an index of the effect of speculative computation, is observable from experimental results as shown in Figure 4.

3.3. Effect of reducing the execution time

When using a stochastic algorithm, it is assumed that the expected value of the computation time and solutions have a relation as that described in Eq. (2). In conventional parallel computing, it is assumed that computation time is reduced by Eq. (1). With the same computation time t_0, the conventional parallel computing is possible to calculate for a time longer by $1/(\alpha/m + (1-\alpha) + \beta)$ times than non-parallel computing. Therefore, the expected value of a solution can be expressed as follows:

$$\mu_p = a \cdot \left(\frac{t}{\alpha/m + (1-\alpha) + \beta} \right)^{-b} + \epsilon. \tag{21}$$

On the other hand, with SSpeC, the expected value μ_m (Eq. (18)) of a solution is obtained using the same computation time t_0. For speculative computation to be more effective

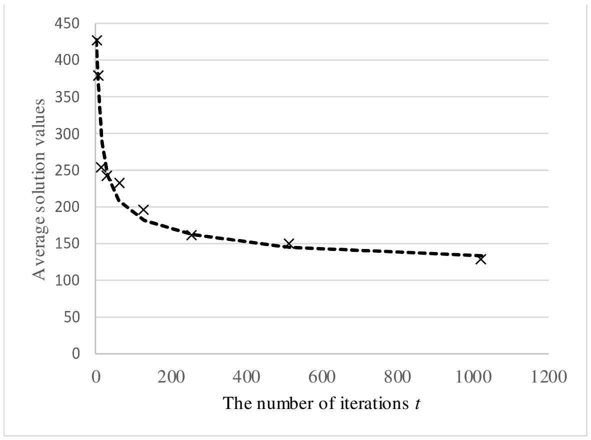

Figure 5. The relationship between the number of iterations t and obtained solution values when the program is executed in a single thread

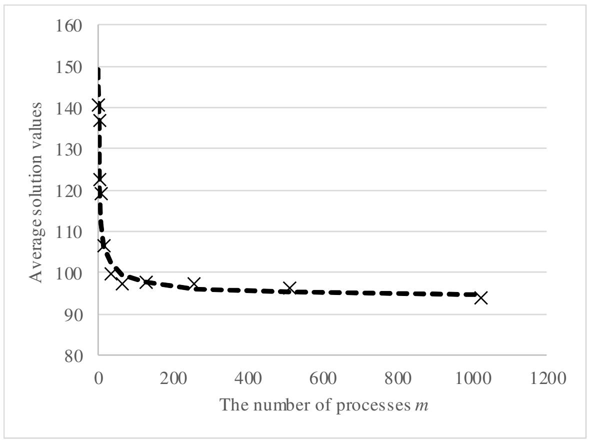

Figure 6. The relationship between the number of processes m and obtained solution values by speculative computation at the fixed 500th iteration point of Figure 5

than the conventional parallelization method, μ_m must be smaller than μ_p.

$$a \cdot \left(\frac{t}{\alpha/m + \beta + 1 - \alpha}\right)^{-b} + \epsilon \quad > \quad \epsilon + m^{-h}(\mu_s - \epsilon). \tag{22}$$

We assume that conventional parallelization can be parallelized in an ideal state. In other words, all parts can be parallelized and the overhead is 0. Under ideal conditions, $\alpha = 1$, $\beta = 0$, and $\epsilon = 0$ for simplicity, Eq. (22) can be transformed as follows:

$$a \cdot (mt)^{-b} \quad > \quad a \cdot t^{-b} \cdot m^{-h}. \tag{23}$$

Thus, the required condition is

$$b < h. \tag{24}$$

When Eq. (24) is satisfied, it is expected that the speculative computation result will be better than that obtained with conventional parallel computing. Note that b and h depend on the problem and algorithm combinations. However, b and h can be easily observed experimentally.

4. Experiment with a Combinatorial Optimization Problem

To quantify the effect of the speculative computation, we used the weighted graph coloring (cost minimization) combinatorial optimization problem with the stochastic algorithm to investigate whether $b < h$ or $b > h$. Here, the topology of the graph is random, the number of nodes is 40, the number of edges is 200, the weight is 1~3, and the number of colors is 3. We used a stochastic iterative improvement algorithm as the base algorithm which transitions to a bad solution with a probability of 0.001. It can also be regarded as the SA algorithm with constant temperature. This algorithm is similar to the one used in the MC molecular simulation using in Section 5.

First, we ran the algorithm using a single thread. Figure 5 shows the relationship between the number of iterations t and solution values. Note that each point is the average of 30 experimental results. This curve can very closely approximate Eq. (2). Note that parameter b can be calculated from this result. An approximate curve is shown in Figure 5. From this curve, in this case, $b = 0.387$ (coefficient of determination $r^2 = 0.969$) in Eq. (2) was obtained. As described in the previous section, from this b, the maximum effect of conventional parallelization can be calculated using Eq.(21) without parallelized execution. This b is compared with the effect obtained by speculative computation in the following experiment.

Next, we executed speculative computation SSpeC, which executed one algorithm in m-parallel with independent random number seeds and independent initial state. The algorithms and parameters used in SSpeC are the same as single-threaded experiments above. After a fixed number of iterations, SSpeC selects the best solution from the obtained solutions. The results are shown in Figure 6. It shows the relationship between the number of processes m and the obtained solution at the fixed 500th iteration point. Since the horizontal axis of Figure 6 is a number of processes m, the point where the horizontal axis is 1 ($m = 1$) is the same as the single thread experimental result of Figure 5. As m is increased, the obtained solution sharply decreases. Here, each point is the average of 30 experimental results; $h = 0.765$ ($r^2 = 0.946$) in Eq.(18) was obtained from the approximate curve shown in the figure. In this experiment, the condition $h > b$ of Eq.(24) was satisfied, although the result was not superlinear. Therefore, speculative computation can be considered to be more effective than conventional parallelization for this problem.

Although $h > b$ in this experiment, when the number of iterations was greater, the solution converged and h tended to decrease. This is because h is considered to be proportional to the dispersion of the solution.

This simple example problem is computable until the optimal solution is found. In a problem for which finding the optimal solution is difficult, the variation of the solutions is large. In such a case, the condition of $h > b$ will be satisfied and the effect of speculative computation will be remarkable.

5. Application of Speculative Computation to MC Molecular Simulation

In the previous section we examined the effect of speculative computation on experimental problems. However, the effect of speculative computation depends on the difficulty of the problem and the base algorithm. Therefore, it is important to conduct experiments with practical problems. In this section, we report the application of speculative computation to an MC molecular simulation.

5.1. Molecular simulation

A molecular simulation is a numerical simulation that calculates the movement of a molecule or analyzes the structure of a molecule. The MD and MC simulations are commonly used in such molecular simulations. In MD simulation, the equations of motion of each atom are solved numerically and the position, speed, energy, and other characteristics of each atom are analyzed. In MC simulation, the state of the molecule in thermal equilibrium is statistically calculated. The calculation principle of the MC simulation is premised on the ergodic Markov process, similar to the case of the stochastic iterative improvement algorithm for a combinatorial optimization problem.

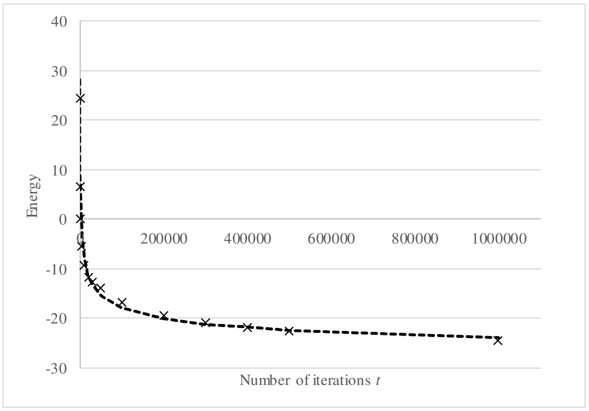

Figure 7. The relationship between the number of iterations t and the energy values obtained by single thread MC simulation

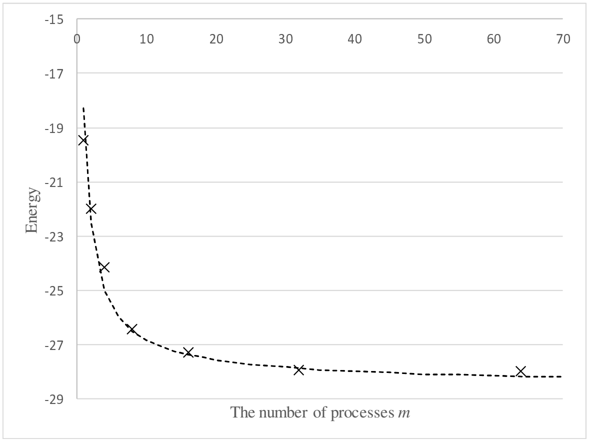

Figure 8. The relationship between the number of processes m and the energy values obtained by speculative computation at the fixed 200,000th iteration point of Figure 7

In MC simulation, computation begins from a certain initial molecular state and reaches a stationary state after multiple state transitions. After reaching the stationary state, the molecule states are sampled while repeating state transitions. The state of a molecule is analyzed by calculating the average of these samples. Here, we use speculative computation to accelerate the process of reaching the near-steady state. In other words, speculative computation solves the problem as a combinatorial optimization problem to find a small energy state.

5.2. Application of speculative computation to MC simulation

Although the speculative computation algorithm assumes abundant computational resources, such as multicore or many-core CPUs, we conducted a pseudo-parallel experiment to analyze the effect of speculative computation on MC simulation. In this experiment, we used the single amino acid potential (SAAP) [16], [17] simulator program as the base MC simulator, and we simulated the structural analysis of chignolin. Chignolin is a peptide comprising 10 amino acids (H-Gly-Tyr-Asp-Pro-Glu-Thr-Gly-Thr-Trp-Gly-OH), which is called the smallest protein. We set the temperature to 300 K in the simulation, and the number of iteration was 200,000.

First, we examined the relationship between the number of iterations and the obtained energy value using a single thread. Figure 7 shows the relationship between the number of iterations t and the average obtained energy values. We obtained $b = 0.235$ ($r^2 = 0.984$) in Eq. (2) from the approximate curve shown in Figure 7.

Next, we conducted experiments of speculative computation in the same way as in the previous section. The relationship between the number of processes m and the average obtained energy values as experimental results at the fixed 200,000th iteration point is shown in Figure 8. As in Figure 6, the horizontal axis of Figure 8 is a number of processes m, and the point where the horizontal axis is 1 ($m = 1$) is the same as the experimental result of the single thread of Figure 7. From the approximate curve shown in the figure, $h = 0.763$ ($r^2 = 0.951$) in Eq.(18) was obtained. Therefore, the condition $b < h$ of Eq.(24) was also satisfied in this molecular simulation. Speculative computation can accelerate MC molecular simulations in this case. However, h is sensitive to problem and algorithm conditions; thus, more detailed experiments are required.

5.3. Discussion

The solution of a stochastic iterative improvement algorithm can be modeled as an ergodic Markov chain. SSpeC executes a stochastic iterative improvement algorithm in parallel. When the frequency distribution of each process solution has converged, can the solution be considered global optimal? From the model of the stochastic process of the Markov chain, this cannot be proved theoretically. However, if Eq. (2) can be assumed and if m is sufficiently large, the converged value can be considered as the near global optimal solution.

Can we confirm this experimentally? As shown in Figure 6, the number of processes m was increased and the obtained solution converged near the optimal solution. However, in the MC molecular simulation, this method is infeasible for determining the minimum energy value. Even with 1,500,000,000 iterations, the solution obtained by running 100 parallelisms did not converge. Approximately 160 h were consumed for computing a single process on an Intel Core i7-4790 CPU @ 3.6 GHz. Note that in MC molecular simulations, even the smallest protein has a very large problem space.

We consider that the speculative computation method is suitable for MC molecular simulations. In this experiment, we used speculative computation to accelerate the simulation process of reaching the near-steady state. The idea of choosing the best result after the computation is finished will be applicable to the simulation itself. Therefore, the following MC simulation experiment was conducted.

Here, the temperature was set to 300 K, the number of iterations was 1,500,000,000, and 20,000 samples were extracted. Chignolin is a peptide whose natural structure is known; thus, the closeness between the obtained sample and the known natural structure was measured using root mean square deviation (RMSD). The distance between the natural structure and the best simulation result obtained from 100 processes was computed to be 1.53Å, and the average distance obtained in 100 processes was 3.79Å. Thus, we confirm that satisfactory results were obtained for both force field energy and RMSD distance. However, theoretical verification of the effect of speculative computation on sample results remains a future problem.

6. Conclusion

In previous studies [6], [7], the effect of speculative computation was calculated theoretically using the state-transition matrix of the Markov process. However, in practical problems, it is difficult to accurately obtain the state-transition matrix. Thus, in this paper, we investigated the effect of the speculative computation method theoretically based on probability theory; moreover, we derived the parameters that can be observed experimentally. The experimental results obtained for the two problems considered in this study demonstrate that it was possible to obtain numerical values for these parameters. In addition, we found that simple speculative computation is effective for such problems. Thus, we confirmed the effectiveness of speculative computation with regards to practical real-world problems. This is the beginning of our speculative computation research. Various algorithms can be considered from the idea of speculative computation. In the future, we would like to improve MC algorithms using this result in order to facilitate the realization of high-speed molecular simulations.

Acknowledgment

We are grateful to Professor Michio Iwaoka, who provided the SAAP MC simulation program for our experiment.

References

[1] R. B. Osborne, *Speculative computation in multilisp*, pp. 103–137. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990.

[2] R. Martí, *Multi-Start Methods*, pp. 355–368. Boston, MA: Springer US, 2003.

[3] A. Sohn, Z. Wu, and X. Jin, "Parallel simulated annealing by generalized speculative computation," in *Parallel and Distributed Processing, 1993. Proceedings of the Fifth IEEE Symposium on*, pp. 416–419, Dec 1993.

[4] K. L. Wong and A. G. Constantinides, "Speculative parallel simulated annealing with acceptance prediction," *IEE Proceedings - Computers and Digital Techniques*, vol. 143, pp. 219–223, Jul 1996.

[5] I. D. Falco, R. D. Balio, E. Tarantino, and R. Vaccaro, "Improving search by incorporating evolution principles in parallel tabu search," in *1994 IEEE Conference on Evolutionary Computation*, pp. 823–828, 1994.

[6] R. Shonkwiler and E. Van Vleck, "Parallel speed-up of monte carlo methods for global optimization," *Journal of Complexity*, vol. 10, no. 1, pp. 64–95, 1994.

[7] X. Hu, R. Shonkwiler, and M. C. Spruill, *Random restarts in global optimization*. Georgia Institute of Technology, 2009.

[8] M. Samadi, A. Hormati, J. Lee, and S. Mahlke, "Paragon: Collaborative speculative loop execution on gpu and cpu," in *Proceedings of the 5th Annual Workshop on General Purpose Processing with Graphics Processing Units*, GPGPU-5, (New York, NY, USA), pp. 64–73, ACM, 2012.

[9] V. Krishnan and J. Torrellas, "Hardware and software support for speculative execution of sequential binaries on a chip-multiprocessor," in *Proceedings of the 12th International Conference on Supercomputing*, ICS '98, (New York, NY, USA), pp. 85–92, ACM, 1998.

[10] Z.-H. Du, C.-C. Lim, X.-F. Li, C. Yang, Q. Zhao, and T.-F. Ngai, "A cost-driven compilation framework for speculative parallelization of sequential programs," *SIGPLAN Not.*, vol. 39, pp. 71–81, June 2004.

[11] L. Yan, "Solving combinatorial optimization problems with line-up competition algorithm," *Computers & Chemical Engineering*, vol. 27, no. 2, pp. 251 – 258, 2003.

[12] C. P. Gomes and B. Selman, "Algorithm portfolios," *Artif. Intell.*, vol. 126, pp. 43–62, 2001.

[13] E. Aarts and J. Korst, *Simulated annealing and boltzmann machines*. New York, NY; John Wiley and Sons Inc., Jan 1988.

[14] Y. Iizuka and K. Iizuka, "Exceeding the efficiency of distributed approximate algorithms enabling by the multiplexing method," in *Knowledge-Based and Intelligent Information and Engineering Systems*, vol. 6883 of *Lecture Notes in Computer Science*, pp. 366–377, Springer Berlin / Heidelberg, 2011.

[15] W. Weibull *et al.*, "A statistical distribution function of wide applicability," *Journal of applied mechanics*, vol. 18, no. 3, pp. 293–297, 1951.

[16] K. Dedachi, T. Shimosato, T. Minezaki, and M. Iwaoka, "Toward structure prediction for short peptides using the improved saap force field parameters," *Journal of Chemistry*, vol. 2013, Article ID 407862, 2013.

[17] M. Iwaoka, N. Kimura, D. Yosida, and T. Minezaki, "The saap force field: Development of the single amino acid potentials for 20 proteinogenic amino acids and monte carlo molecular simulation for short peptides," *Journal of computational chemistry*, vol. 30, no. 13, pp. 2039–2055, 2009.