PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT. © ICOT, 1992

Folding Simulation using Temperature Parallel Simulated Annealing

Makoto Hirosawa,* Richard.J.Feldmann,† David Rawn,‡
Masato Ishikawa,* Masaki Hoshida,*
George Micheals†

hirosawa@icot.or.jp

Abstract

We applied *temperature parallel* simulated annealing to the biological problem of folding simulation. *Water-counting* is introduced to formulate folding simulation as an optimization problem. Nobody has ever solved the folding simulation problem. We cannot obtain biologically significant consequences either. However, from the viewpoint of the evaluation value of the folding simulation, we observed the effectiveness of parallel computing.

1 Introduction

Folding simulation uses a computer to simulate the process of protein formation from its stretched state to its native folded state. This research topic has held the interest of biologists for a quarter of a century and has never been solved. No researche has been able to reach the native folded state by folding simulation. Three of the authors(Feldmann, Rawn and Micheals) have been interested in formulation for protein folding. They introduced *the water-counting model*, which requires solution by computer.

Meanwhile, the other three the authors(Hirosawa, Ishikawa and Hoshida) have studied the application of Multi-PSI [Nakajima *et al.* 1989] parallel inference machine to biological problems. A first attempt was made to the problem of multiple alignment [Ishikawa *et al.* 1991] using *temperature parallel* simulated annealing [Kumura and Taki 1990]. It was so successful that other biological applications were sought.

*Institute for New Generation Computer Technology(ICOT)
†National Institutes of Health
‡Towson University

As the requirements of both partners matched, we combined efforts to conduct collaborative research. The purpose of this research was to investigate the applicability of the optimization algorithm, temperature parallel simulated annealing, to folding simulation and to evaluate the effectiveness of the water-counting model.

The concept of folding simulation is explained in the second section and the water-counting model and its computational formulation are introduced in the third section. Then, temperature parallel simulated annealing is explained in the fourth section. Finally, the simulation results are shown in the fifth section.

2 What is folding simulation?

2.1 Biological background of folding simulation

Proteins are biological substances and they are essential to the existence of all creatures, from humans to the AIDS virus. A protein is a linear chai of amino acids. It consists of 20 kinds of amino acids. The structure of protein is determined by the order of the amino acids in the sequence. The structure of protein is closely related to its function. Therefore, it is very important to know the structure of the protein.

Even now, it is very difficult to determine the structure of a protein. X-ray crystallography and NMR(Nuclear Magnetic Resonance) can be used to determine structure. But the former method can only be utilized when crystalization of protein is succeessful, and this crystalization is very difficult to do. The latter method can be adopted when the size of the protein is small. Both require plenty of time from months to a year.

On the other hand, we can determine the order of amino acids in the sequence of protein extremely easier than we can determine a structure of a protein. A technique for determining the sequence of a protein has been established. That is why folding simulation is important and necessary.

Folding simulation simulates, by computer, the process of protein formation from its stretched state to its native folded state. Before simulation starts, information on the order of the amino acids is provided.

2.2 Folding simulation as an optimization problem

Folding simulation is a research topic that has fascinated hundreds of theoretical bio-chemists for a quarter of a century. The molecular dynamic method is, theoretically, able to solve folding simulation problem. The method precisely simulates the movement of each atom driven by kinetic forces. However, it requires such huge amounts of computational time that actual folding simulation problems cannot be solved (it can simulate pico-second movements of a protein whereas the whole folding process takes a few seconds or more). To make the computational time tractable, we have to seek effective approximation methods.

In each approximation method, abstract representation (*e.g. the amino-acid ball* which represents all atoms in an amino acid as a single ball) and the limited structure state (*e.g.* limited location or angle) are often introduced. We can regard such an approximation method as combinatorial optimization, because each discrete state is evaluated by a properly-defined potential energy to be minimized and effective transition between states is devised.

One of the most frequently employed approximation methods is lattice representation [Ueda *et al.* 1978] [Skolnick and Kolinsky 1991], which restricts the position of amino acids in 3-dimensional lattice cells. Although the lattice representation can remarkably reduce computational time for folding simulation, no significant result had been produced until recently. That is partly because the lattice formulation might not be good enough to simulate the folding process, and partly because the computational power required might have still been too big.

The work by Skolnick and his co-researchers is the first research that did not poorly reproduce the native structure of protein by the folding simulation. However, the parameterization he employed to reproduce the native structure has drawn criticism.

3 Computational Formulation of Folding Simulation

We introduced a water-counting model to approximate folding simulation concisely and to formulate folding simulation as an optimization problem based on the model. The water-counting model employs a lattice representation for protein and water.

3.1 Concept of water-counting model

Coming back to basic biological knowledge, we have sought a simulation method that requires the most minimal parameterization possible. Then, we found the water-counting model as a biological model.

In 1958, I.M. Klotz recognized that the folded structure of a protein depends upon its interaction with water [Koltz 1958]. At about the same time, W.Kauzmann showed that the *hydrophobic effect* provides the principle driving force for protein folding [Kaumann 1959].

Hydrophobicity is a measure that represents the degree to which amino acids don't favor water. Amino acids that favor water are called hydrophilic amino acids, and those that don't are called hydrophobic amino acids. Hydrophobic force is caused by the above tendency of amino acids. Since many biologists today, if not all, still recognize hydrophobic force as a primary force, we simplified folding simulation by employing hydrophobic force without using any other kind of force.

Next, we investigated the origin of hydrophobic force. We concluded that the binding and detaching of water to and from amino acids produces hydrophobic force. We can interpret the global minimum energy of protein in terms of the number of water molecules bound to proteins. Because the energy is calculated by the number of water molecules around amino acid, we coined it the water-counting model.

3.2 Representation of Protein

We will describe a way to represent protein on a 3-dimensional lattice. In lattice cells, any place protein is not present, water will fill.

As described earlier, proteins are linear chains of amino acids. Each amino acids is composed of two parts, namely, a main chain and a side chain. Main chains form the backbone of protein. Side chains of amino acids determine the properties of the amino acid.

The main chain of an amino acid serves to connect adjacent amino acid. The relative location between two adjacent amino acids is like the move that a knight in chess makes, but on a 3-dimensional lattice (Figure 1), $(\pm3, \pm1, \pm1)$. Every main chain of amino acid occupies 27 ($= 3^3$) lattice cells.

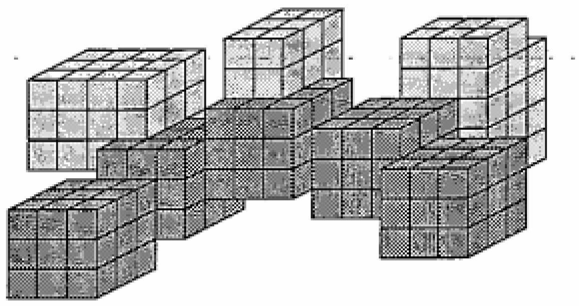

Figure 1: Representation of a part of protein: main chains(shaded) and side chains(unshaded)

Each of the twenty kinds of amino acids has different side chains (Table 1). For example, their volume (the number of lattice cells occupied) and hydrophobicity [Janin 1979] differ.

3.3 Evaluation of State

The energy of states are evaluated in the following formula.

$$E(Energy) = \sum_{m}^{sidechains}(Water\ Count_m - 1) \times Hydrophobicity_m$$

$$Water\ Count_m = \frac{Number\ of\ adjacent\ cells\ (of\ side\ chain)\ occupied\ by\ other\ amino\ acids}{The\ number\ of\ adjacent\ cells\ of\ the\ side\ chain}$$

In the first formulas, the terms from hydrophobic amino acids are negative and those from hydrophilic amino acid are positive. The more the absolute value

Amino acid	A	C	D	E	F	G	H
Hydrophobicity	0.3	0.9	−0.6	−0.7	0.5	0.3	−0.1
Volume	12	16	20	32	52	0	40

Amino acid	I	K	L	M	N	P	Q
Hydrophobicity	0.7	−1.8	0.5	0.4	′−0.5	−0.3	−0.7
Volume	48	60	48	40	28	28	40

Amino acid	R	S	T	V	W	Y
Hydrophobicity	−1.4	−0.1	−0.2	0.6	0.3	−0.4
Volume	68	16	28	36	68	56

Table 1: Characteristics of amino acid side chains: each letter signifies one of 20 amino acids, for example, E signifies glutamic acid.

of the hydrophobicity of an amino acid is, the greater its contribution to the energy is.

The energy can be reduced both by increasing the amount of water around the hydrophilic amino acid and by reducing the amount of water around the hydrophobic amino acid. The minimization of energy has the effect of inviting hydrophobic amino acids toward the center of the protein where there is less water and to oust hydrophilic amino acids to the surface of the protein where water is abundant.

3.4 Transition between States

As a transition from one state to another, we introduce two classes of transition. One is rotational transition and the other is translational transition (Figure 2).

Rotational transition is the move that proteins probably take in actual folding processes. We first focus on one amino acids and select which side of the protein to rotate (in the figure, the right side is selected). Then, by regarding a connection line between the focused amino acid and its adjacent amino acid (in the figure, the adjacent amino is on the left) as an fixed axis, the selected side of the protein is rotated.

Translational transition is the moving of proteins that is done for computational convenience. As with rotational transition, one amino acid is focused on and the side to move is selected. Then, the adjacent amino acid of the selected protein is moved and other amino acids on the selected side are moved translationally. (the direction to translate is specified by the move of the adjacent amino acid).

After a new state is created by the transition selected, a collision check is executed. If, in the next possible state, there is no multiply occupancy of any lattice cell by different parts of the protein, this state is acceptable. Otherwise, the state is discarded and new transitions are tested until some that is accepted is found.

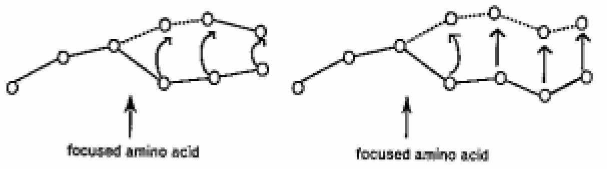

Figure 2: Rotational transition (left) and Translational transition (right)

4 Temperature Parallel Simulated Annealing

In the proceeding section, we formulated folding simulation as the problem to search for the minimum energy in a solution space. We employed temperature parallel simulated annealing as an algorithm to find a global optimal solution. Temperature parallel simulated annealing is an algorithm that can circumvent a scheduling problem of simulated annealing (SA), by introducing the concept of parallelism in temperature.

In this section, SA is explained firstly, then temperature parallel SA is introduced.

4.1 SA

SA is a stochastic algorithm used to solve complex combinatorial optimization problems [Kirkpatrick 1983]. It searches for a global optimal solution in a solution space without being captured in local optima.

SA simulates the annealing process of physical systems using a parameter, *temperature*, and an evaluation value, *energy*. At high temperatures, the search point in the solution space jumps out of local energy minimum. At low temperatures, the point falls to the nearest local energy minimum.

An outline of the SA algorithm is as follows. Given an arbitrary initial solution x_0, the algorithm generates a sequence of solutions $\{x_n\}_{n=0,1,2,...}$ iteratively, finally outputting x_n for a large enough value of n. In each iteration, the current solution x_n is randomly modified to get a candidate x'_n for the next solution, and the variation of the energy $\Delta E = E(x'_n) - E(x_n)$ is calculated to evaluate the candidate. When $\Delta E \leq 0$, the modification is good enough to accept the candidate: $x_{n+1} = x'_n$. When $\Delta E > 0$, the candidate is accepted with probability $p = \exp(-\Delta E/T_n)$, but rejected otherwise: $x_{n+1} = x_n$, where $\{T_n\}_{n=0,1,...}$ is a *cooling schedule* (a sequence of temperatures decreasing with n).

Because solution x_n is distributed according to the Boltzmann distribution at temperature T, the distribution converges to the lowest energy state (optimal solution) as the temperature decreases to zero (Figure 3). Thus, one might expect SA to be capable of providing the optimal solution, in principle. It is well-known that the cooling schedule has great influence on SA performance. This is where the *cooling schedule problem* arises.

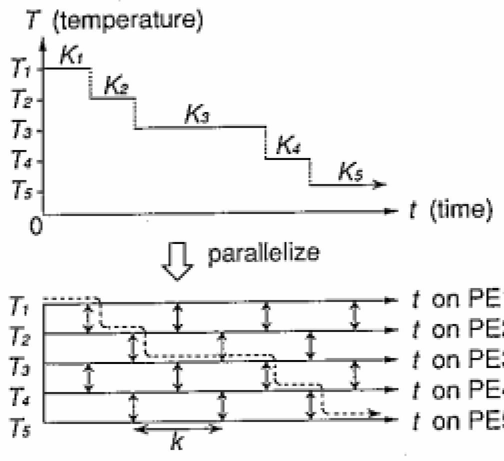

Figure 3: Ordinary SA and temperature parallel SA

4.2 Temperature Parallel SA

The basic idea behind the algorithm is to use parallelism in temperature [Kumura and Taki 1990], to perform annealing processes concurrently at various temperatures. The algorithm automatically constructs an appropriate cooling schedule from a given set of temperatures (Figure 3). Hence, it partly solves the cooling schedule problem.

The outline of the algorithm is as follows. Each processor maintains one solution and performs the annealing process concurrently at a *constant* temperature that differs between processors. After every k annealing steps, each pair of processors with adjacent temperatures performs a *probabilistic exchange of solutions*. Let $p(T,E,T',E')$ denote the probability of the exchange between two solutions: one with energy E at temperature T and the other with energy E' at temperature T'. This is defined as follows:

$$p(T,E,T',E') = \begin{cases} 1 & \text{if } \Delta T \cdot \Delta E < 0 \\ \exp(-\frac{\Delta T \cdot \Delta E}{TT'}) & \text{otherwise} \end{cases}$$

$$\text{where} \qquad \Delta T = T - T', \qquad \Delta E = E - E'.$$

The probability has been defined such that solutions with lower energy tend to be at lower temperatures. Hence, the solution at the lowest temperature is expected to be the best solution so far. The cooling schedule is invisibly embedded in the parallel execution.

The temperature parallel algorithm has advantages other than the dispensability of the cooling schedule. We can stop the execution at any time and examine whether a satisfactory solution has already been obtained.

The algorithm of temperature parallel SA is implemented as a tool kit. When we want to solve some problem using temperature parallel SA, if we use the tool kit, all we have to do is to write a program that just corresponds to the problem.

5 Experiment and Discussion

We selected flavodoxin, whose structure is known, as the protein to simulated. This protein is of a medium size and has 138 amino acids. We ran the folding simulation program using temperature parallel SA on Multi-PSI using 20 processors over 10 days. This corresponds to 30,000 cycles. We also ran the folding simulation program using *simple parallel* SA in 30,000 cycles, also with 20 processors.

The simple parallel SA is a naive combination of sequential SAs: every available processor has one solution and anneals it sequentially using a *distinct* sequence of random numbers. All resultant solutions are compared with each other and the best one, the one with the minimum energy value, is selected as a solution for the algorithm .

5.1 Experimental result

The minimum energy versus the cycles of simulation of those two algorithms is plotted in Fig.4. In the figure, the result using *sequential* SA, ordinary SA, is also plotted. Its energy is the average of energy obtained by sequential SAs in the simple parallel SA.

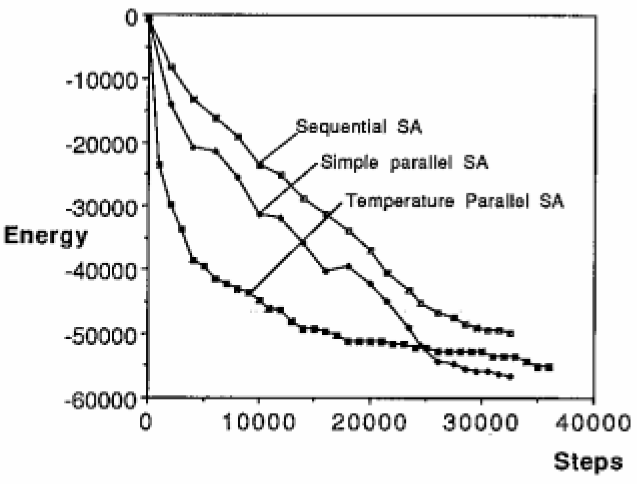

Figure 4: Energy history of folding simulation

One of the structure of flavodoxin produced by the program is shown in Figure 5. Unfortunately, its structure is not similar to the structure of real flavodoxin. However, a favorable tendency ,where hydrophobic amino acids are inside the structure while hydrophilic amino acids are outside the structure, was observed.

5.2 Discussion

The effectiveness of the water-counting model will be evaluated first, then the effectiveness of the temperature parallel SA as an optimization method for practical problems will be evaluated.

The structure of the flavodoxin produced was not similar to its real structure. However, this doesn't necessarily indicate a defect in the water-counting model. We, instead, think that the result is due to insufficiency of transitions we introduced.

The rough structure of protein, especially that of small protein, can be reproduced by global transition

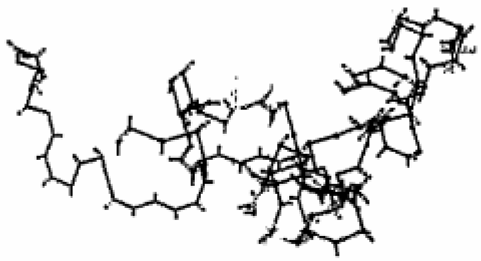

Figure 5: Result structure of folding simulation (flavodoxin)

that is like rotational transition and translational transition. There is little collision among amino acids in the path from the stretched state to the roughly formed structure. However, a fine protein structure is rarely reproduced by global transitions alone due to the collisions.

We think that the local transition modes that can avoid collision should be incorporated to reproduce the native structure with collision check. We are planning to introduce a local transition, kink mode [Skolnick and Kolinsky 1991]. We think that the necessary mode of transition must be incorporated before we can evaluate the effectiveness of the water-counting model.

Next, we evaluate the effectiveness of temperature parallel SA as an optimization method by using Figure 4. Readers who are familiar with SA should consult Appendix.

We made the following observations from this energy profile in consideration of the above points.

1. Two kinds of parallel SAs made better results within a fixed time than sequential SA. This is simply the effect of multiple processors.

2. Up to the middle stage of simulation, temperature parallel SA is always better than simple parallel SA. This is because temperature parallel SA can produce optimal solutions as that time.

3. Two kinds of parallel SAs have almost the same final energy value.

Figure 4 shows the tendency for energy of all methods to be minimized further after the completion of a specified cycle of simulation. Only simulation by temperature parallel SA can be resumed without rescheduling. Because two kinds of parallel SAs are almost the same, we think that temperature parallel SA is more advantageous than simple parallel SA.

Simulated annealing is most effective when states generated at higher temperatures can cover nearly all the solution space. In the case of folding simulation, this is hard to do it. We are now engaged in trying to restrict the solution space of simulated annealing by knowledge and/or heuristics to the extent that the solution space can be covered by simulated annealing.

6 Summary

We studied folding simulation as an application of parallel simulated annealing. This program was written in KL1 and was executed on the parallel inference machine Multi-PSI. As the biological model the water-counting model that uses lattice representation and only hydrophobic interaction between amino acids was selected.

The structure of flavodoxin produced by program is not appropriate from a biological point of view. This suggests that the program requires further improvements. The kink mode of transition is one candidate to incorporate.

However, the insight was gained from the point view of computer science, namely evaluation of temperature parallel simulated annealing. The result using temperature parallel SA had almost the same final energy value (which is much better than that obtained by sequential SA) as the result using simple parallel SA. In consideration of the dispensability of rescheduling when further optimization is necessary, temperature parallel SA was proved to be advantageous.

The other thing we learnt was that a module that restricts the solution space of folding simulation is required. We think knowledge engineering must be employed to do this, and also that KL1 is suitable for use.

Acknowledgment

The authors would like to especially thank Y. Totoki of IMS for his programming and experimentation with the folding simulation. Without his endeavors which were conducted through what should have been his winter vacation, this paper couldn't have been written. We would also like to thank Kouichi Kimura, the founder of temperature parallel SA, for valuable discussions.

We also thank Dr. Uchida and Dr. Nitta for their support in this international collaboration.

References

[Kumura and Taki 1990] Kimura, K. and Taki, K. (1990) Time-homogeneous parallel annealing algorithm. *Proc. Comp. Appl. Math. (IMACS'91),* **13**, 827–828.

[Nakajima *et al.* 1989] Nakajima, K., Inamura, Y., Ichiyoshi, N., Rokusawa, K. and Chikayama, T. (1989) Distributed implementation of KL1 on the Multi-PSI/V2. *Proc. 6th Int. Conf. on Logic Programming.*

[Ishikawa *et al.* 1991] Ishikawa, M., Hoshida, M., Hirosawa, M., Toya, T., Onizuka, K. and Nitta, K. (1991) Protein Sequence Analysis by Parallel Inference Machine. *Information Processing Society of Japan, TR-FI-23-2,* (in Japanese).

[Gierasch and King 1990] Gierasch, L.M and King, J(ed) (1990) Protein folding. *American association for the advance of science.*

[Skolnick and Kolinsky 1991] Skolnick, J. and Kolinski, A. (1991) Dynamic Monte Carlo Simulation of a New Lattice Model of Globular Protein Folding, Structure and Dynamics. *Journal of Moleculer Biology Vol. 221 no.2,* 499-531.

[Ueda *et al.* 1978] Ueda, Y., Taketomi, H. and Go, N. (1978). Studies on protein folding, unfolding and fluctuations by computer simulation. A three dimensional lattice model of lysozyme. *Bilpolymers Vol.17* 1531-1548.

[Koltz 1958] Koltz,I.M. (1958) *Science Vol.128,* 825-

[Kauzmann 1959] Kauzmann (1959) *Advances in Protein Chemistry, Vol.14, no.1.*

[Janin 1979] Janin, J. (1979) Surface and side volumes in globular proteins. *Nature(London) Vol.277,* 491-492.

[Kirkpatrick 1983] Kirkpatrick, S., Gelatt, C.D. and Vecci, M.P. (1983) Optimization by simulated annealing. *Science, vol.220, no.4598.*

Appendix

Readers should pay attention to the following possibility when they discuss the result of Figure 4.

1. The two energy histories obtained by the two parallel SA algorithms might include the influence of statistical fluctuation, because each parallel algorithm was experienced only once. The sequential algorithm, however, was done twenty times and each point in the history represents the average energy value.

2. All SA procedures may be quick quenched instead of annealed, because the number of steps at each temperature, 1500, would be relatively small against the size of the solution space. If so, temperature parallel SA is disadvantageous for obtaining good energy in a short time, because not all processors in temperature parallel SA will necessarily do quick quenching; some processors may often do real annealing.

3. All SA proceduresy may not reach any minima in the solution space, because every decline in energy history is not sufficiently saturated.