Implementation of Protein Tertiary Structure Prediction System with NetSolve

Yusuke Tanimura
Graduate Student of Engineering
Doshisha University
tanisuke@mikilab.doshisha.ac.jp

Keiko Aoi
Graduate Student of Engineering
Doshisha University
aoi@mikilab.doshisha.ac.jp

Tomoyuki Hiroyasu
Department of Engineering
Doshisha University
tomo@is.doshisha.ac.jp

Mitsunori Miki
Department of Engineering
Doshisha University
mmiki@mwind.jp

Yuko Okamoto
Department of Theoretical Studies
Okazaki National Research Institute
okamotoy@ims.ac.jp

Jack Dongarra
Department of Computer Science
University of Tennessee
dongarra@cs.utk.edu

Abstract

In this study, the protein tertiary structure prediction systems on the Grid are proposed for progress of the bioinformatics. The prediction is mainly performed by the protein energy minimization. However, this method has many iterated calculation of the protein energy in most cases. To use the Grid as the large-scale computing environment would be valuable for this system. In the system, Parallel Simulated Annealing using Genetic Crossover (PSA/GAc) is a minimization engine and NetSolve is a basic tool to use the Grid. In this study, two types of implementations are prepared. The first naive implementation of the system has a critical overhead due to large communication delay over the Internet. The second system, asynchronous Crossover model, improves the performance in the second implementation. The details of the system and the experimental results solving C-peptide are shown as an example of Grid application.

1. Introduction

The protein is a natural substance and an essential part in the organic phenomenon. Since the biological function of the protein is specified by the 3-D structure, the protein folding problem, how it folds into this structure, is one of the great challenges in science today. The design of new medicine, creating an artificial protein and analysis of genetic disease are achieved from the resolution of that problem. The natural conformation of the protein in real world is corresponded to free-energy minimal state and it is predictable by computational search with optimization algorithm. In this study, Parallel Simulated Annealing using Genetic Crossover (PSA/GAc)[6] is applied to the energy minimization. PSA/GAc is a combined model with Simulated Annealing (SA)[7] and Genetic Algorithm (GA)[5]. In this algorithm, plural processes of SA are running separately. GA's crossover is performed in exchange of the transient solution that has been found by each SA. In the previous study[6], PSA/GAc has applied for minimizing the energy of protein structures on distributed memory computes like PC Clusters. The implementation models of PSA/GAc were discussed. Then, it was found that the master-slave model can keep high parallel efficiency and search capability. However, it was also found that huge computing power to search in the large conformation space is necessary for the energy minimization of the protein that has more than several hundreds of amino acids. Therefore, huge computational resources are necessary for minimizing protein structures. On of the solutions for preparing these resources is the Grid technology. Those resources could be provided from several computer centers and shared with many scientific application users.

In this study, PSA/GAc systems that are work on the Grid are proposed. In the proposed system, the Grid is a basic infrastructure to execute PSA/GAc and the NetSolve is a tool to build it. The SA calculation service is prepared as NetSolve server on remotes and registered to NetSolve agent. NetSolve client can get service information by query

to the agent. GAs crossover is implemented as a part of the client and it works after the client receives the results of SA calculation on remote servers. In this study, two types of implementation models of PSA/GAc using NetSolve are prepared. In the nave master-slave model, the crossover operation is performed after all the processes of SA are returned to the client. However, to submit a SA execution from the client is a RPC request. The client must invoke the RPC many times in the Master-slave model. The RPC has some overhead and it would be very large over the Internet. In the asynchronous master-slave model, the crossover operation is performed just after two processes of SA are returned to the client. In this model, it is expected that the overhead of Grid RPC is small. By applying these systems to minimize the energy of C-peptide, the network overheads of systems and search capability are compared between these two models.

2. Application Development on The Grid

2.1. The Grid

The Grid is a terminology from the Power Grid and it is distinguished as future computing infrastructure that is flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources[3]. It enables the Grid users to access huge sharable resources and useful public information from anywhere at anytime. To build the Grid computing environment, the middleware is highly needed to establish secure and efficient system examined about fault tolerance, authentication, authorization, scheduling, resource monitoring, and several communication methods. Globus Toolkit[2], Ninf[10], NetSolve[1] and Condor[8], which are middlewares to solve some of the above issues. Most application programmers can develop their own application easily with those middleware without particular knowledge for the Grid.

2.2. GridRPC Programming Framework

The GridRPC is one of the programming models that have a Remote Procedure Call (RPC) mechanism tailored for the Grid[14]. It provides a simple API to invoke RPC for coarse-grained parallel tasking. The Ninf and the NetSolve mainly support this GridRPC API and their developer's teams contributed their experiments and results to the Advanced Programming Model Research Group of the Global Grid Forum[4].

2.3. The NetSolve/GridSolve

The proposed system has been developed using the GridRPC API of NetSolve. NetSolve is a client-server system which provides remote access to hardware and software resources through a variety of client interfaces, such as C, Fortran and Matlab[1].

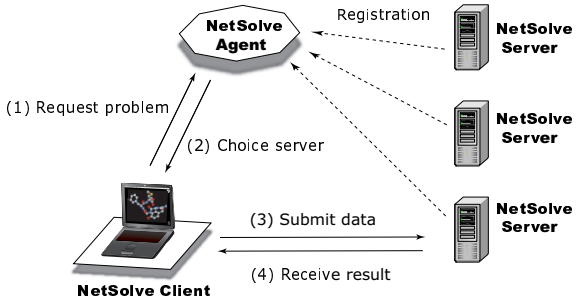

Figure 1. The NetSolve System

The NetSolve system consists of three entries, as illustrated Figure 1. The Client may have a user interface that sometimes includes interactive operation. The RPC call is invoked from the client to the Server. The server executes its service according to the client request. The Agent keeps information for all servers and provides it to the client. A typical RPC to NetSolve involves several steps, as follows:

Step 1 The client queries the agent for an appropriate server that can execute desired function.

Step 2 The agent returns a list of available servers, ranked in order of suitability.

Step 3 The client attempts to contact a server. The client then sends the input data to the server.

Step 4 Finally the server executes the function on behalf of the client and returns the results.

In NetSolve, there are several types of overhead to invoke the call: cost to query to the agent, communication cost between the client and the server, and cost of NetSolve internal process.

3. Implementation of PSA/GAc using NetSolve

3.1. PSA/GAc Overview

Parallel Simulated Annealing using Genetic Crossover (PSA/GAc) is the optimization method. In this algorithm, several SA processes are running in parallel. After some steps, these processes are stopped and the temporal solutions among SAs running in parallel by the genetic crossover[6]. In case one SA obtains an optimum value of one of the design variables, the crossover operation works to transfer the value to the other SA. It is expected that the

optimum solution is derived in the early stage. In this algorithm, the total number of SAs running in parallel is defined as population size and the solution of each SA is corresponded to one individual of GA. The following steps are the flow of PSA/GAc.

Step 1 Initial individuals are generated in GA.

Step 2 Each individual is given to one SA as an initial searching point. SA is executed in parallel in the order of generating a new solution, accept the solution, and cooling a temperature.

Step 3 When annealing has been done for the specified d steps (Crossover interval), the transient solution of SAs come back to GA. The pairs are generated randomly from those solutions. The number of pairs is a half number of the individuals.

Step 4 The crossover is applied to a pair and two children are generated. The crossover method is explained later.

Step 5 The best two individuals are selected from four individuals; two parents and two children. These two individuals are new initial searching points for the next SA.

Step 6 The operations from Step 4 to Step 5 are applied to all the pairs.

Step 7 The steps between two to six are iterated until the terminal criterion is satisfied.

3.2. Synchronous Master-slave Model

The first naive system is implemented as Synchronous Master-slave model of PSA/GAc in the framework of GridRPC. SA calculation is executed on the server and the crossover is performed on the client. In general, the master-slave model requires large calculation on the server side in order to ignore communication delay between master and slave. To apply it to this case, SA on the server is useful because SA includes many iterated calculation of the protein energy. Figure 2 describes the implementation of the synchronous master-slave model of PSA/GAc. NetSolve servers work as a slave and NetSolve client does as a master. The genetic crossover is operated after all SAs have finished for specified steps. A periodical synchronization over all servers is necessary for this algorithm. The SA program is previously prepared on the server using NetSolve IDL and the client calls it with GridRPC API of NetSolve. The search procedure is shown as follows:

Step 1 The client prepared initial individuals in GA.

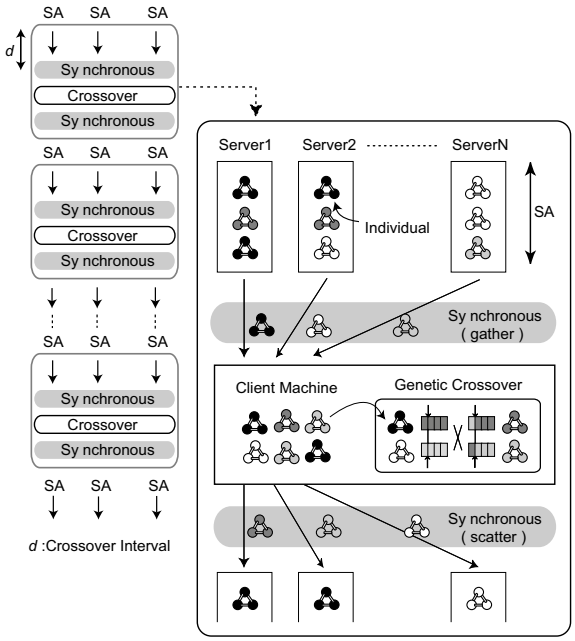

Figure 2. Synchronous Master-slave Model

Step 2 The client queries a SA service to the agent and then the client invokes RPC to an introduced server from the agent. The input data that is an individual information is transferred to the server and the SA is started.

Step 3 SA is executed independently on each server for the specified d steps.

Step 4 After all SAs return the results to the client, the client decides pairs randomly and operates the genetic crossover.

Step 5 Steps between two to four are repeated until the terminal criterion is satisfied.

The synchronous master-slave model can be implemented without any change of the basic PSA/GAc. Since the RPCs are called with non-blocking, SAs are executed in parallel. However, the non-blocking call is processed sequentially and there is still small delay. This overhead of 1 RRC is 0.05 [sec] between 2 nodes of the same PC cluster networked with 100 Mbps Ethernet. The total overhead of the RPC is in proportion to (number of individuals -1) $\times 0.5$.

3.3. Asynchronous Master-slave Model

Asynchronous Master-slave model is an implementation to hide communication delay and dispersion of time to finish RPCs. It is predictable that latency is very high on the

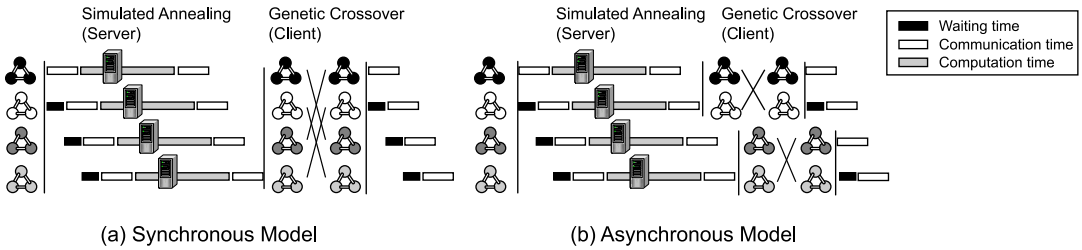

Figure 3. Difference of Synchronous Model and Asynchronous Model

Grid because the client is far away from the server. Each server is running on a varied node that has different performance. The asynchronous master-slave model solves these issues to omit the synchronization at the genetic crossover. A pair of the crossover is decided in the order of time, in which the results are returned. The search procedure of this model is described as follows:

Step 1 The client prepared initial individuals in GA.

Step 2 The client queries a SA service to the agent and then the client invokes RPC to an introduced server from the agent. The input data that is individual information is transferred to the server and the SA is started.

Step 3 SA is executed independently on each server for the specified d steps.

Step 4 When each SA finished the specified d steps, the results are returned to the client's receive archive.

Step 5 If there are two individuals in the receive archive, the first two individuals are selected as a pair of the genetic crossover and two children are generated.

Step 6 The best two individuals are elected and stored in the send archive.

Step 7 If there are some individuals in the sending archive, those are transferred to the server that is assigned by the agent. The next SA process is started.

Step 8 Steps from three to seven are repeated until the terminal criterion is satisfied.

Figure 3 describes the difference of the synchronous master-slave model and the asynchronous master-slave model of PSA/GAc. By using the asynchronous master-slave model, the total execution time would be reduced, however the search model is different from original one. Therefore, there is a possibility that the accuracy of the derived solution is worse than that of synchronous model. In the following section, these points are discussed through the numerical experiments.

4. Comparison of Synchronous Master-slave Model and Asynchronous Master-slave Model

4.1. Overview of Numerical Examples

In the first experiment, the synchronous master-slave model and the asynchronous master-slave model are compared in the total execution time and the searching capability.

In this numerical experiment, C-peptide is a target protein to solve with PSA/GAc. C-peptide consists of 13 amino acids, *Ly+*, *Gl-*, *Thr*, *Ala*, *Ala*, *Ala*, *Ly+*, *Phe*, *Glu*, *Ar+*, *Gln*, *Hi+*, and *Met*. From the experiments by Okamoto et al[12], the structure has minimal energy when seven amino acids shape alpha helix, in the gas-phase simulation based on ECEPP/2[9, 11, 15]. The minimal energy is about -42.2 kcal/mol. In this paper, the alpha helix is shaped when the dihedral angles (ϕ_i, ψ_i) by more than 3 series of amino acids are ($-60 \pm 45°$, $-50 \pm 45°$). The design variables of energy minimization are 26 dihedral angles of the main chain and 38 dihedral angles of the side chain. In the SA to solve C-peptide, 1 MCsweep (described at next subsection) has 64 metropolis judgments.

The function to derive the energy of protein is based on the energy parameters of ECEPP/2. This function is the gas-phase simulation. The dihedral angles of backbone and side chains are applied as design variables. Values of the dihedral angles are in the range of $[-180°, 180°]$. Each dihedral angles is generated and given the accept criterion sequentially, and then the temperature is cooled. This series of operation is defined as a Monte Carlo Sweep (MCsweep) in this paper.

In this experiment, the initial dihedral angles are generated randomly. In generating process, the next state is produced in the neighborhood using probabilistic model of uniform distribution. Equation 1 gives the range of the neighborhood. *Total # sweeps* equals to the number of MCsweeps at the end of SA. *# sweep* shows the current number of MCsweeps.

Table 1. Specification of PC Cluster

	# Proc	Processor	Memory
Doshisha (Server)	5	PentiumIII 800 MHz	256 MB
Doshisha (Server)	20	PentiumIII 600 MHz	256 MB
Doshisha (Server)	5	PentiumIII 500 MHz	512 MB
Doshisha (Agent)	1	PentiumIII 500 MHz	512 MB
Doshisha (Client)	1	PentiumIII 600 MHz	256 MB

Proc: Number of processors

Table 2. Specification of Internet Environment

	# Proc	Processor	Memory
Tennessee(Server)	10	PentiumIII 900 MHz	256 MB
Tennessee(Server)	10	PentiumIII 550 MHz	512 MB
Doshisha(Server)	3	Pentium4 2400 MHz	512 MB
Doshisha(Agent)	1	PentiumIII 1100 MHz	256 MB
Doshisha(Client)	1	Pentium4 2400 MHz	512 MB

Proc: Number of processors

Table 3. Throughput between Client and Server

Netperf Client	Netperf Server	Throughput
Doshisha (Client)	Doshisha (Server)	94.05Mbps
Doshisha (Server)	Doshisha (Client)	94.12Mbps
Doshisha (Client)	Tennessee (Server)	0.49Mbps
Tennessee (Server)	Doshisha (Client)	0.86Mbps

Table 4. Parameters of PSA/GAc in Experiment of Synchronous and Asynchronous Master-slave Model

Initial temerature	2.0 (1000K)
Population size	2, 4, 8, 16
Crossover interval	10, 100 MCsweep
Cooling rate	0.998
Range size	$180^\circ \rightarrow (180 \times 0.3)^\circ$
Number of trials	5

$$\begin{cases} max = 180^\circ - \dfrac{180^\circ \times 0.7 \times \# \; sweep}{Total \; \# \; sweeps} \\ min = -max \end{cases} \quad (1)$$

The experiments are performed on both the PC cluster system and the Internet environment.

In the PC Cluster system, all nodes are connected to the local network of 100 Mbps Ethernet, while each node has different performance showed in Table 1

Table 2 shows the specification of the internet environment. The difference from the PC cluster is that some nodes for the server are far away from the client. The site of the client is Doshisha University and the site of the some servers is University of Tennessee. The throughput between them are showed in Table 3.

Table 4 shows the parameters of PSA/GAc in this experiment. Cooling schedule and neighborhood range are based on the papers [13]. The crossover intervals are 10 MCsweep and 100 MCsweep, which affect the results significantly. In case the interval is 10 MCsweep, the total RPC count is (number of individuals $\times 150$). In case of 100 MCsweep, the total RPC count is (number of individuals $\times 15$) so that the aveage number of the energy calculation is 1500 MCsweep per individual in each SA.

4.1.1 Total Execution time

Total execution time is calculated between spawning the client process and halting it after all RPCs have been done. The average results of five trials are shown in Figure 4. This figure shows the execution time (sec) with the horizontal one does the number of individuals (as same as the number of parallel execution in this case) for PSA/GAc. The parallelization of PSA/GAc is not for reduction of execution time but for improvement of searching capability. It means that the execution time must be constant as the number of parallilization is getting higher. However, the overhead of RCP system prevents it. Figure 4 shows that the execution time of asynchronous model is lower than that of the synchronous one in all the cases. This result leads that the asynchronous model is very effective. At the same time, while the difference is small in the experiment at the cluster, it is very large in the experiment at the internet environment. Figure 4 also shows that the difference of the execution time is remarked when the crossover interval is short. The lowest result comes from the case where the synchronous model with 10 MCsweep interval was performed at the internet environment. On the other hand, the asynchronous model with enough interval is clearly useful on the internet environment.

4.1.2 Details of Execution Time

Figure 5 shows the profile of the execution time of the asynchronous master-slave model and RPC calls to the total execution time are illustrated.

When the crossover interval is short like 10 MCsweep, the most part of the total execution time is the communication and waiting time to finish RPC. From this result, it is found that the crossover interval is expected to be longer. On the other hand, the longer crossover interval might reduce the search capability. This affection is explained in the

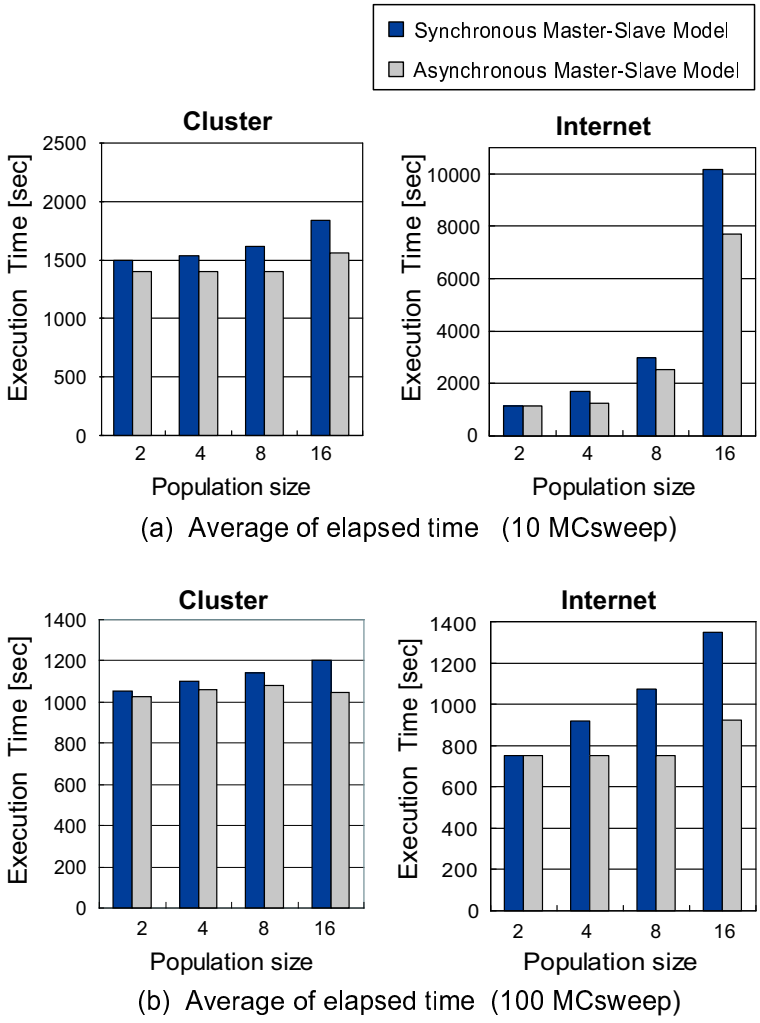

Figure 4. Comparison of Total Execution Time

next section.

4.1.3 Accuracy of Solution

The asynchronous model has the different searching capability from the original PSA/GAc due to the different selection method of the crossover. A pair is not selected random from the total population in the asynchronous model. It depends on the performance of the assigned server to execute SA. In all the cases of Figure 6, the derived results by asynchronous model are almost the same as those of synchronous model. From these results, the asynchronous model is also acceptable due to the similar searching ability as the synchronous one.

4.2. Effect of Crossover Interval

In the previous experiment, there are two types of the crossover intervals. It was founded that the longer interval is effective to reduce the ratio of overhead in total execution. The interval, however, affects the search capability.

Figure 7 shows the success ratio to find the sufficiently low energy to shape well-known C-peptide structure in ECEPP/2 and the minimum energy found by each trial, in case to use the synchronous master-slave model of PSA/GAc . In Figure 7, PSA indicates that each SA is independently executed at the same time without any exchange of searching information due to no crossover. The parameters of the results are shown in Table 5.

The results in Figure 7 describes that there is an optimum interval for the genetic crossover. In case of execution with (16 individuals $\times 6000$) MCsweep, the optimum

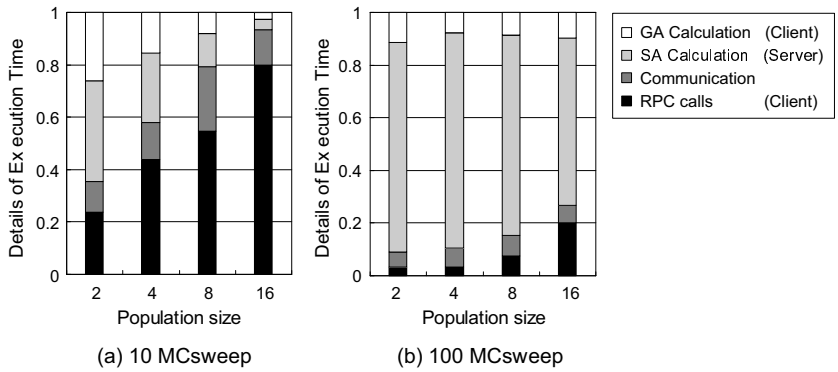

Figure 5. Detailed Execution Time of Asynchronous Master-slave Model

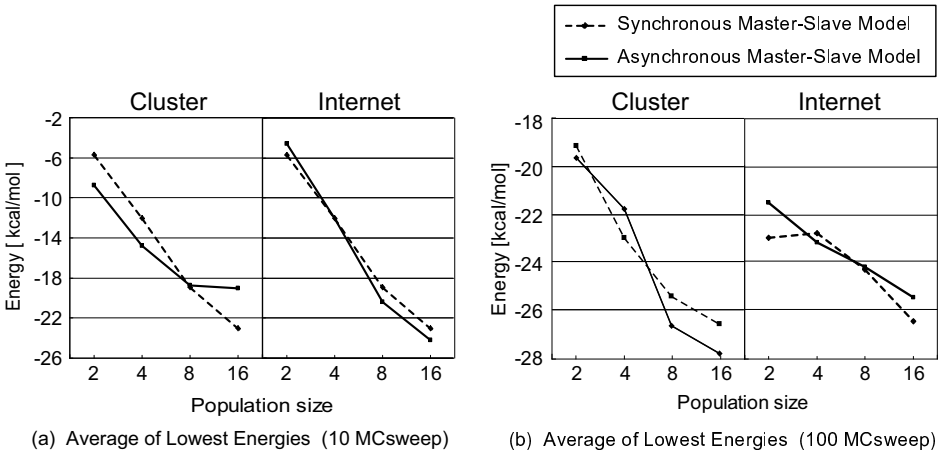

Figure 6. Comparison of Searching Capability

Table 5. Parameters of PSA/GAc in Experiment for Crossover Interval

Initial temerature	2.0 (1000K)
Population size	16, 32, 64
MCsweep per each SA	6000
Crossover interval	2, 4, 8, 16, 32, 64, PSA
Cooling rate	0.998
Range size	$180^\circ \rightarrow (180 \times 0.3)^\circ$
Number of trials	50

crossover interval is 32 MCsweep. As the same case of execution with (32×3000) MCsweep, the optimum is four MCsweep. In case of (64×1500) MCsweep, the optimum is two MCsweep. The total MCsweep of all SAs is constant in those cases. These results describe that it is important to set an appropriate optimum crossover interval. Nevertheless, it causes a tradeoff between searching capability and parallel efficiency.

5. Conclusion

The protein tertiary structure prediction system with NetSolve is proposed, developed and experimented on the Grid. PSA/GAc is used not only for superior prediction engine but also for easiness to implement a master-slave model using GridRPC API. To execute SA, which is a part of the main calculation, on the slave side is valuable for the Grid where communication delay might be high. There are two types of implementation models; the synchronous master-slave model and the asynchronous master-slave model. The latter model can hide the network and RPC overhead during maintaining the parallel efficiency and the accuracy of the solutions. On the other hand, the experiment shows that there is some difficulty to set an appropriate the crossover interval. This study describes an example to implement the Grid application using the GridRPC based system and it re-

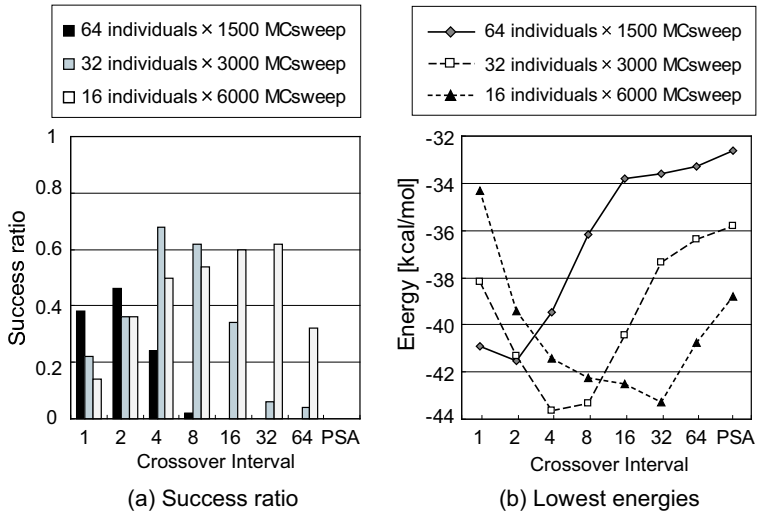

Figure 7. Effect of Genetic Crossover Interval

veals how overhead should be hidden in the master-slave model.

References

[1] H. Casanova and J. Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. *Supercomputer Applications and High Performance Computing*, 11(3):212–223, 1997.

[2] I. Foster and C. Kesseleman. Globus: A Metacomputing Infrastructure Toolkit. *Supercomputer Applications*, 11(2):115–128, 1997.

[3] I. Foster and C. Kesseleman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[4] Global Grid Forum. http://www.gridforum.org/.

[5] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[6] T. Hiroyasu, M. Miki, M. Ogura, and Y. Okamoto. Examination of Parallel Simulated Annealing using Genetic Crossover. *Journal of Information Processing Society of Japan*, 43(SIG7(TOM6)):70–79, 2002.

[7] S. Kirkpatrick, C. D. Gelatt, and J. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[8] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor – A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems*, pages 101–111, 1998.

[9] F. Momany, R. McGuire, A. Burgess, and H. Scheraga. Energy parameters in polypeptides. vii. geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occurring amino acids. *J. Phys. Chem.*, 79:2361–2381, 1975.

[10] H. Nakada, M. Sato, and S. Sekiguchi. Design and Implementations of Ninf: towards a Global Computing Infrastructure. *Future Generation Computing Systems, Metacomputing Issue*, 15:649–658, 1999.

[11] G. Nemethy, M. Pottle, and H. Scheraga. Energy parameters in polypeptides. 9. updating of geometric parameters, nonbonded interactions and hydrogen bond interactions for the naturally occurring amino acids. *J. Phys. Chem.*, 87:1883–1887, 1983.

[12] Y. Okamoto, M. Fukugita, T. Nakazawa, and H. Kawai. α-Helix folding by Monte Carlo simulated annealing in isolated C-peptide of ribonuclease A. *Protein Engineering*, 4(6):639–647, 1991.

[13] Y. Okamoto, T. Kikuchi, and H. Kawai. Prediction of Low-Energy Structures of Met-Enkephalin by Monte Carlo Simulated Annealing. *CHEMISTRY LETTERS*, pages 1275–1278, 1992.

[14] K. Seymour, H. Nakada, S. Matsuoka, J. Dongarra, C. Lee, and H. Casanova. GridRPC: A Remote Procedure Call API for Grid Computing. Technical Report ICL-UT 02-06, Innovative Computing Laboratory, 2002.

[15] M. Sippl, G. Nemethy, and H. Scheraga. Intermolecular potentials for crystal data 6. determination of empirical potentials for 0-h—o=c hydrogen bonds for packing configurations. *J. Phys. Chem.*, 88:6231–6233, 1984.