Computational Intelligence and Applications(2002) p.p.149-154

Temperature Parallel Simulated Annealing with Adaptive Neighborhood for Continuous Optimization Problem

Mitsunori MIKI†, Tomoyuki HIROYASU†, Masayuki KASAI††, Keiko ONO†† Takeshi JITTA††

† Department of Knowledge Engineering, Doshisha University,
†† Graduate Student, Doshisha University,
Doshisha University
Kyo-tanabe, Kyoto, 610-0321, JAPAN
Email: mmiki@mail.doshisha.ac.jp

1 Introduction

There is a strong incentive to parallelize the computation for optimization problems since it requires many iterations of analysis. Especially, new approaches to optimization problems such as genetic algorithms and simulated annealing, which are very effective for solving complicated optimization problems with many optima, require tremendous computational power. Consequently, parallelization of these new optimization methods, which sometimes are called heuristic search methods[1], is very important.

It was Kirkpatrick et al. who first proposed simulated annealing, SA, as a method for solving combinatorial optimization problems[2]. It is reported that SA is very useful for several types of combinatorial optimization problems[3]. The advantages and the disadvantages of SA are well summarized in [4]. The most remarkable disadvantages are that it needs a lot of time to find the optimum solution and it is very difficult to determine the proper cooling schedule. To determine the proper cooling schedule, many preparatory trials are needed. When the cooling schedule is not proper, the guarantee of finding optimum solution is lost.

There are two approaches to shorten the calculation time in SA. One is determining the cooling schedule properly. SA with the proper cooling schedule can provide an optimum solution quickly. This approach is well reported by Ingber[4]. The other approach is to perform SA on parallel computers. Because of the rapid progress of parallel computers, there are several studies with this approach[5]. Among these studies, the temperature parallel simulated annealing (TPSA)[6], which was called the time-homogenous parallel annealing[7] before, is one of the algorithms that can overcome the cooling schedule problem. TPSA is an algorithm that can be carried out on parallel computers easily and does not require any cooling schedule. These are remarkable advantages. So far, TPSA has been applied to LSI allocation problems[6], traveling salesman problems[8], graph partition problems[7] and so on. However, there are very few studies that focus on continuous optimization problems. Therefore, the effectiveness of TPSA in continuous problems has not been clear.

In this study, a new TPSA approach that can be applied to continuous optimization problems is proposed. In the proposed approach, the SA that Corana developed and TPSA are combined and the neighborhood range is determined adaptively. The approach is called temperature parallel simulated annealing with adaptive neighborhood (TPSA/AN).

2 Temperature Parallel Simulated Annealing

Comparing to sequential SA, there are more sophisticated algorithms that have proven that parallel probabilistic exchange of information gathered from processors annealing at constant but different temperatures can increase the overall rate of convergence. Kimura and Taki called this algorithm temperature parallel simulated annealing (TPSA)[6]. In TPSA, each processor performs a sequential SA with a constant temperature for the whole annealing time, while the different temperatures are assigned to different processors. Two solutions in two processors with adjacent temperatures are exchanged with a certain probability at an interval of annealing time. The basic concept of the TPSA is shown schematically in Fig. 1.

The important features of TPSA are as follows. (a) The cooling schedule is determined automatically because solutions decide their temperatures by themselves. (b) After getting solutions, when these solutions are not satisfied, TPSA can be restarted to get better solutions.

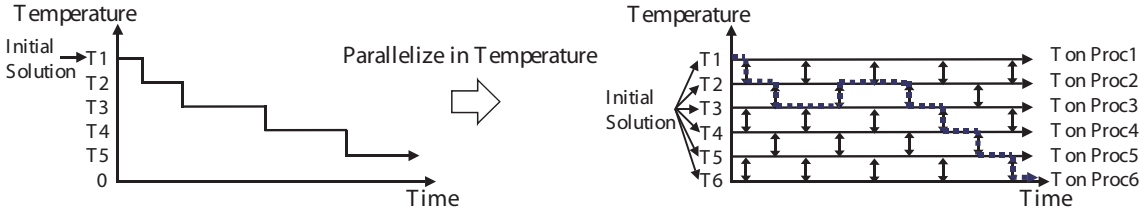

Figure 1: Sequential SA and temperature parallel SA

When the energy of a solution at higher temperature is lower than that at lower temperature, the solutions are always exchanged. Otherwise, when the energy of a solution at higher temperature is higher than that at lower temperature, the solutions are exchanged in accordance with the probability that is derived from the differences in temperature and energy. This probability function is defined in equation (1). By this method, the solutions that have low energy tend to concentrate to the lower temperature.

$$P_{EX}(T, E, T', E') = \begin{cases} 1 & if \Delta T \cdot \Delta E < 0 \\ \exp\left(-\frac{\Delta T \cdot \Delta E}{T \cdot T'}\right) & otherwise \end{cases} \tag{1}$$

where T is temperature, E is energy, and the prime means the state of the adjacent temperature. ΔT and ΔE are the differences in the temperature and the energy between two adjacent temperatures.

As mentioned before, in each processor of a parallel computer, one or several sequential SA with constant temperatures are performed. The acceptance probability is defined by the Metropolis criterion that is shown in equation (2), where x is a design variable.

$$\begin{aligned} P_{AC}(T, E, E') &= \begin{cases} 1 & \Delta E \leq 0 \\ \exp\left(-\frac{\Delta E}{T}\right) & otherwise \end{cases} \\ \Delta E &= E(x_{new}) - E(x_{old}) \end{aligned} \tag{2}$$

3 SA for continuous optimization problems

Simulated Annealing (SA)[2] has been proposed in the area of combinatorial optimization[3]. However, SA has been also used in the area of continuous optimization problems which had a lot of local minima[4][9][10][11][12][13].

The definitions of the neighborhoods of solutions in the design space in SA are different between discrete and continuous optimization problems. For combinatorial optimization problems, the neighborhood of a solution is defined by a small change in the combination (e.g. 2-change neighborhood for the traveling salesman problems[14]). On the other hand, the neighborhood of a solution in continuous optimization problems is defined by a distance in the design space, which can be handled more easily than in combinatorial one.

Thus, for continuous optimization problems, it is important to determine the neighborhood range for generating a next point in a problem space[4]. If the neighborhood range is fixed to a constant range, the range should be determined individually for a particular problem. When the range is too large, it becomes difficult to get an accurate solution, while it takes much time to get better solution when the range is too small.

Therefore, for continuous optimization problems, there are several methods with adjustable neighborhood range. Many practitioners use novel techniques to narrow the range as the search progresses. For example, Boltzmann annealing method uses Gaussian distribution whose standard deviation is a squared root of the temperature[4]. Fast annealing method uses the Cauchy distribution[11]. In these methods, the neighborhood range is large if the temperature is high, while the neighborhood range is small if the temperature is low. However these methods are not effective for searching in problem spaces, because they do not use the information about objective functions.

There are several methods using the information about objective functions. Corana's SA[10] uses the information, measured by a rate between accepted and rejected moves. VFSR[9] uses the pseudo sensitivities of the objective function. Dekker & Aarts's SA[13] uses local search procedure (e.g. steepest descent, quasi-Newton).

4 Temperature Parallel Simulated Annealing with Adaptive Neighborhood (TPSA/AN)

In this paper, we propose Temperature Parallel Simulated Annealing with Adaptive Neighborhood(TPSA/AN) which is the extension of TPSA by using the Corana's SA[9] for continuous optimization problems. The difference

between the conventional sequential SA for combinatorial optimization problems and TPSA/AN is that TPSA/AN has a procedure for exchanging solutions between different constant temperatures and a procedure for adjusting the neighborhood range. The exchange of solutions and the adjustment of the neighborhood range are executed at certain transitions. In this algorithm, the distribution for generating a next point x' from current point x is as follows:

$$x_i{}' = x_i + rm \tag{3}$$

where r is a random number generated in the range [-1, 1]; m is the neighborhood range. The algorithm in this paper has the same parameter m for each design values, while Corana's SA has a different parameter for each design value. If the next point x' lies outside the definition domain of an objective function, our algorithm will generate a new point again. The neighborhood range m is varied for adaptive search using equations (4), (5), (6) and (7):

$$m_{new} = m_{old} * g(p) \tag{4}$$

$$g(p) = 1 + c\frac{p - 0.6}{0.4} \quad , \text{ if } p > 0.6 \tag{5}$$

$$g(p) = \left(1 + c\frac{0.4 - p}{0.4}\right)^{-1} \quad , \text{ if } p < 0.4 \tag{6}$$

$$g(p) = 1 \quad , \quad \text{otherwise} \tag{7}$$

where c is a multiplying factor for adjusting the neighborhood range ; p is a rate between accepted and rejected moves, and it is calculated from the following equation:

$$p = n/N \tag{8}$$

where N is a given number of transitions; n is the number of accepted moves in interval N. In this paper, parameter c is set to 2, following the Corana's paper.

In this algorithm, if the number of the accepted moves increases, the neighborhood range will be enlarged adaptively by equations (4) and (5). If the number of the rejected moves increases, the neighborhood range will be reduced adaptively by equations (4) and (6). With this adaptation, the rate between the accepted and the rejected moves is adjusted to be in the range from 0.4 to 0.6, and this makes the computational effort on each parallel processor of a parallel computer effective.

5 Parallel Implementation

In TPSA/AN, inter-processor communications occur at the exchange of solutions in different processors at a certain annealing interval. Therefore, the communications are not frequent, and then TPSA/AN is very suitable for parallel processing.

We use a PC-cluster with 8 processors, which are connected by the Fast Ethernet. PVM is used for a communication interface. The CPUs are Pentium II, 233MHz.

One temperature is assigned to one process in PVM. Therefore, one process is running in one processor as the number of temperature stages is less than 8, but multiple processes are running in one processor as the number of temperature stages is greater than 9. In this study, 4 processes are running in one processor when the number of temperature stages is 32.

6 Minimization of Standard Test Function

To examine the performance of TPSA/AN, we minimize the standard test functions, such as the Rastrigin function[15], the Griewangk function[15], and the Shekel functions[16]. In this study, the functions has two design variables, and they are expressed by equations (9), (10) and (11).

$$f_{\mathrm{R}}(\vec{x}) = (N \times 10) + \left[\sum_{i=1}^{N} \left(x_i^2 - 10\cos(2\pi x_i)\right)\right] \tag{9}$$

$$f_{\mathrm{G}}(\vec{x}) = 1 + \sum_{i=1}^{N} \left(\frac{x_i^2}{4000}\right) - \prod_{i=1}^{N} \left(\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) \tag{10}$$

$$f_{\mathrm{S}}(\vec{x}) \quad = \quad -\sum_{j=1}^{5} \left((\vec{x} - \vec{a}_j)^T (\vec{x} - \vec{a}_j) + c_j \right)^{-1} \tag{11}$$

where a_{ij} and c_j are shown in Table 1.

Table 1: Values for a_{ij} and c_j

i	a_{ij}		c_j
1	4	4	0.1
2	1	1	0.2
3	8	8	0.2
4	6	6	0.4
5	3	7	0.4

In this study, to evaluate the effectiveness of TPSA/AN, a sequential SA and TPSA are also implemented. The sequential SA and TPSA whose neighborhood ranges are fixed are called SA/FN and TPSA/FN, respectively. The sequential SA whose neighborhood range is changed adaptively is called SA/AN. The parameters are summarized in Table 2 for the three functions. The parameters are a maximum temperature (for the sequential SA, this means its starting temperature), a minimum temperature (for the sequential SA, this means its ending temperature), the number of annealing cycles, the Markov length, the cooling rate and the exchange intervals. The parameters are determined from the experience. The number of iterations in TPSA means the number of iterations in each temperature. Therefore, from the point of view for a single processor, the sequential SA performs annealing 32 times longer than in TPSA.

Table 2: Parameters for SA and TPSA

Functions	Rastrigin		Griewangk		Shekel	
Algorithms	**SA**	**TPSA**	**SA**	**TPSA**	**SA**	**TPSA**
Number of Processes	1	32	1	32	1	16
Max.(Initial) temperature	10		20		0.8	
Min.(Final) temperature	0.01		0.01		0.01	
Number of iterations	10240×32	10240	30720×32	30720	80×16	80
Markov length	10240	-	30720	-	80	-
Cooling rate	0.80025	-	0.726	-	0.640414	-
Exchange interval	-	32	-	32	-	4
Neighborhood adjust interval	8		8		8	

7 Results of Numerical Experiments

TPSA/AN is carried out for the minimization of the three standard test functions to discuss the effectiveness of the method.

Figures 2 through 4 show the energies of the optimum solutions obtained by SA and TPSA with respect to the neighborhood range for the three test functions. The neighborhood ranges are set to 0.01, 0.05, 0.l, 0.5, 1.0, 5.0 and the adaptive one. The interval used for adjusting the neighborhood range is 8 annealing steps. The results are shown as the medians of 10 trials since the energy have much different values. Since the relationship between the values of the energy and the objective function is linear, the point that has the minimum value of the energy has the minimum value of the objective function. The total number of annealing steps in SA and TPSA are fixed in order to compare the results.

These results are summarized as follows.

Comparing SA/AN(Adaptive Neighborhood) with SA/FN(Fixed Neighborhood), the performance of SA/AN is worse than the SA/FN with the optimum fixed neighborhood range, but it shows moderate performance in the whole. Therefore, SA/AN has an advantage in handling a neighborhood range since the optimum neighborhood range is not determined unless we conduct many preliminary experiments.

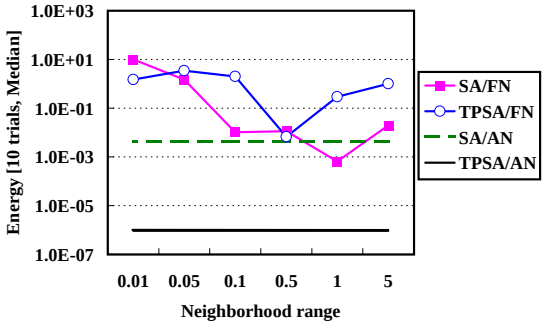

Figure 2: The performance of the various method for the Rastrigin function.

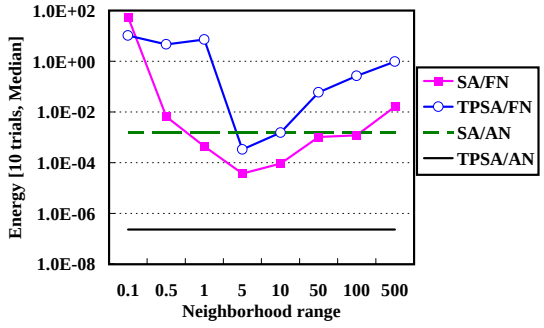

Figure 3: The performance of the various method for the Griewangk function.

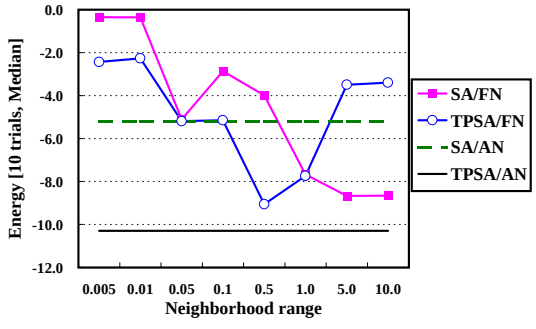

Figure 4: The performance of the various method for the Shekel function.

Comparing TPSA/AN with the other methods, it provides a remarkable performance. The performance is mach better than SA/AN, and is better than the performance of SA/FN with the optimum fixed neighborhood range. Therefore, the proposed method is found to be very effective for various optimization problems. Using this method one does not need to determine the neighborhood range and the searching performance is very high as well.

It is interesting that the performance of TPSA/FN is worse than SA/FN. This result suggests that the parallelization of SA is very difficult. But, using the adaptive neighborhood range, the parallelization of SA yields very good performance.

From these results, it is clear that the determination of the appropriate neighborhood range is very important to obtain good solutions. Also, it is recognized that the best values of the neighborhood ranges for SA/FN and TPSA/FN are different.

The effectiveness of TPSA/AN can be examined by comparing the histories of the energies of the solutions which arrived at the minimum temperature, as shown in Fig. 5, where the Rastrigin function is minimized for various fixed ranges of neighborhood and the adaptive neighborhood.

The solution with a small neighborhood range (0.1) was not improved at the early stage of search, and it reached the local minimum. The solution with a large neighborhood range (5.0) was improved at the early stage, but it was not improved later. The solution with a proper neighborhood range (0.5) converged to a good solution.

On the other hand, the solution with the adaptive neighborhood reached the best solution among these. The adaptive neighborhood has both good global and local search abilities, which are inherently contrary each other. However, this advantage appears only for TPSA, but not for sequential SA. The performance of SA/AN is moderate among various fixed neighborhood ranges.

8 Conclusions

In this study, the temperature parallel simulated annealing with adaptive neighborhood (TPSA/AN) that is able to solve optimization problems whose design variables are continuous is proposed. The effectiveness of TPSA/AN is investigated through the Rastrigin function. The following results are derived in this study.

1) The neighborhood range affects the search abilities of SA and TPSA in continuous optimization problems. Therefore, the determination of the best neighborhood is very important.

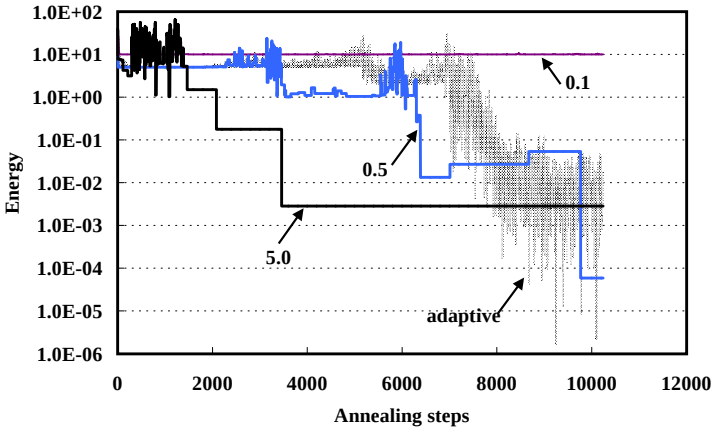

Figure 5: The histories of the energies of solutions for various fixed neighborhood and adaptaive neighborhood.

2) The optimum obtained by TPSA/F with the best neighborhood range is worse than the one obtained by SA/F with the best neighborhood range.

3) The optimum solution obtained by TPSA/AN is better than the one obtained by SA/F, SA/AN and TPSA/F.

4) TPSA/AN has high ability of global searching and quick convergence to the global optimum. Therefore, TPSA/AN is a powerful algorithm for continuous optimization problems.

References

[1] Beasley, J., Dowsland, K., Glober, F., Laguna, M., Peterson, C., Reeves, C. R. and Söderberg, B., Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications, 1993

[2] Kirkpatrick, S., Gelett Jr. C. D., and Vecchi, M. P., Optimization by Simulated Annealing, Science, 220(4598), 1983, 671-680

[3] Aarts, E. and Korst, J., Simulated Annealing and Boltzmann Machines, John Wiley & Sons, 1989

[4] Ingber, L., Simulated Annealing: Practice versus Theory, J. of Mathl. Comput. and Modelling, 18(11), 1993, 29-57

[5] Holmqvist, K., Migdalas, A., and Pardalos, P. M., Parallelized Heuristics for Combinatorial Search, in Parallel Computing in Optimization, Migdalas, A. et al. eds., Kluwar Academic Publishers, 1997, 269

[6] Konishi, K., Taki, K. and Kimura, K., Temperature Parallel Simulated Annealing Algorithm and Its Evaluation, Trans. on Information Processing Society of Japan, 36(4), 1995, 797-807 (in Japanese)

[7] Kimura, K. and Taki, K., Time-homogeneous Parallel Annealing Algorithm, The 13th IMACS World Congress of Computation and Applied Mathematics, 1991

[8] Konishi, K., Yashki, M. and Taki, K., An Application of Temperature Parallel Simulated Annealing to the Traveling Salesman Problem and its Efficient Implementation on the Distributed Memory Parallel Machine, 1996 Joint Symposium of Parallel Processing, 1996, 153-160 (in Japanese)

[9] Ingber, L., Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison, Mathematical and Computer Modeling, 16(11), 1992, 87-100

[10] Corana, A., Marhesi, M., Martini, C. and Ridella, S., Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm, ACM Trans. on Mathematical Software, 13(3), 1987, 262-280

[11] Szu, H. and Hartley, R., Fast Simulated Annealing, Physics Letters A, 122(3,4), 1987, 157-162

[12] Rosen, B., Functional Optimization based on Advanced Simulated Annealing, IEEE Workshop on Physics and Computation, PhysComp92 (Dallas, Texas), 1992, 289-293

[13] Dekkers, A. and Aarts, E., Global optimization and simulated annealing, Mathematical Programming, 50, 1991, 367-393

[14] p.7 of the reference [4]

[15] Whitley, D., Mathias, K., Rana, S. and Dzubera, J., Evaluating Evolutionary Algorithms, Artificial Intelligence, 85, 1996, 245-2761

[16] Madsen, K., Testing branch-and-bound methods for global optimization, Technical Reports 2000 , 2000