Neighborhood Parallel Simulated Annealing

Keiko Ando[1], Mitsunori Miki[2], and Tomoyuki Hiroyasu[2]

[1] Graduate School of Knowledge Engineering and Computer Science, Doshisha Univ., Kyoto, Japan. ando@mikilab.doshisha.ac.jp

[2] Department of Knowledge Engineering and Computer Science, Doshisha Univ., Kyoto, Japan. {mmiki@mail, tomo@is}.doshisha.ac.jp

1 Introduction

Simulated annealing(SA) is a method for solving combinatorial optimization problems. SA was formulated by Kirkpatrick et al., and its algorithm imitates physical annealing. It is very useful for solving several types of optimization problems with nonlinear functions and multiple local optima[1, 2, 3].

The advantages and disadvantages of SA are well summarized in [4].The most significant disadvantages are the difficulty in determining the appropriate parameters, and the long calculation time for determining the optimum solution. The neighborhood range and temperature are the important parameters in SA. In combinatorial optimization problems, the neighborhood range can be determined by replacing two adjoining elements in solutions. Thus, a neighborhood range is decided automatically for every problem; therefore, the temperature schedule becomes the most important factor. On the other hand, when SA is applied to continuous optimization problems, a neighborhood range can be freely decided in the range of the Eucledean space. However, the appropriate neighborhood range is greatly dependent on the landscape of the objective function. Therefore, the control of the neighborhood range becomes very important when applying SA to a continuous optimization problem, but it is difficult to detremine a neighborhood range automatically in this case.

Some research has been carried out wherein the neighborhood range was adjusted adaptively according to the landscape of the objective function. Corana's method[6] controls the neighborhood range according to a landscapes maintaining an acceptance ratio of 0.5. The authors proposed a new method called SA/AAN wherein the appropriate neighborhood range can be determined based on the arbitrary acceptance ratio. These methods do not require the tuning of their parameters for many problems; however, the target values of the acceptance ratio are necessary.

In this research, we propose a parameter-free SA method called Neighborhood Parallel Simulated Annealing (NPSA). This method determines the neighborhood range automatically and does not require the acceptance ratio in order to adjust a neighborhood range.

2 NPSA

2.1 The NPSA concept

In NPSA, parallel processing is performed and neighborhood ranges are assigned during each process. The neighborhood ranges are self-adjusted through the exchange of neighborhood ranges between processors. Such an algorithm enables the exchange of the neighborhood ranges between processors based on the energy of the solutions. The processor with a low energy is assigned a small neighborhood range, so that the solution can search locally. On the other hand, the processor with a high energy is assigned a large neighborhood range, so that the solution can avoid the local minima. Thus, the processors cooperate and adjust neighborhood ranges as the search proceeds.

2.2 The NPSA algorithm

The NPSA algorithm is pictorially depicted in Fig. 1. The process followed in NPSA is described below.

1. Parallel processing is performed.
2. A different neighborhood is assigned during in each process.
3. SA is excecuted in parallel in each processor.
4. At every cooling cycle, the energy values(evaluation value) are gathered in one process.
5. The energy values are sorted, and the rank is determined based on the energy.
6. The neighborhood range is assigned according to the rank as follows. Processor B is ranked 1; therefore, this processor's neighborhood is determined to be the smallest. Processor D is ranked 4; therefore, this processor's neighborhood is determined to be the biggest. The same logic is applied to the remaining cases.
7. Steps 3 to 6 are repeated.

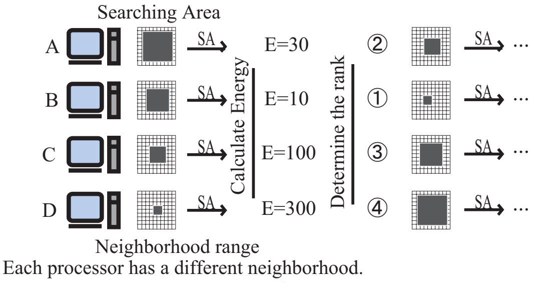


Fig. 1. NPSA algorithm

3 Effectiveness varification of the proposed method

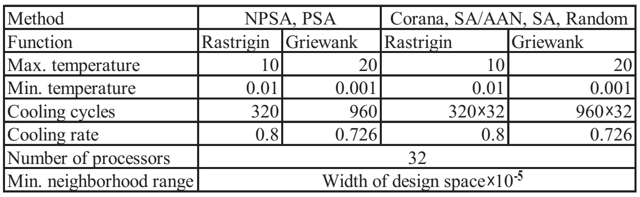
3.1 Optimization problems

In order to exmine the search performance of the proposed method, two standard test functions are used, namely the Rastrigin and the Griewank functions. The optimum solutions are located at the origin for the Rastrigin and

Griewank functions, and the function values are 0, and its function value is also 0. Two, three, and five dimensions of these functions are used as test functions.

The parameters are indicated in Table 1. The number of both cooling cycles and parallel processors were set to 32. Thirty-two parallel processors are used in both NPSA and PSA, and the annealing step was set to the same value as that in sequential methods by setting the cooling cycle to 1/32 times.

Table 1. Parameters



Method	NPSA, PSA		Corana, SA/AAN, SA, Random	
Function	Rastrigin	Griewank	Rastrigin	Griewank
Max. temperature	10	20	10	20
Min. temperature	0.01	0.001	0.01	0.001
Cooling cycles	320	960	320×32	960×32
Cooling rate	0.8	0.726	0.8	0.726
Number of processors	32			
Min. neighborhood range	Width of design space×10^{-5}			

3.2 Results and discussion

The minimum energy value obtained using the Rastrigin function is indicated in Fig. 2, while that using Griewank function is indicated in Fig. 3. The comparison between the Random Search, Conventional SA(SA), Corana, SA/AAN, and NPSA methods in Fig. 2 and 3 clearly demonstrates the proposed method's(NPSA) exceedingly superior performance on all dimensions.

We now consider the effectiveness of NPSA by compareing the energy histories of PSA, which is the standard method of SA, and NPSA, which demonstrates a superior performance. It is clear from Fig. 4 and 5 that NPSA yields a better solution and arrives at the optimum solution faster than PSA. The neighborhood range selected can be considered as the reason for this.

This difference in performance stems from the variation of the neighborhood range. Fig. 6 shows the history of the neighborhood ranges. PSA maintains range 1, which is the best range arrived at through extensive preliminary experimentation. On the other hand, NPSA utilizes the new adaptive mechanism for assigning the neighborhood range. In Fig. 6, the solutions trapped in a local minima in the first stage; therefor, the neighborhood range become quite large. In the middle stage, the solution has converged on the global minimum, and the neighborhood range has become increasingly smaller for local search. In the final stage, this mechanism can search the global optimum area. Thus, NPSA can adaptively vary the neighborhood range as search process proceeds, thereby facility effective searching.

4 Conclusion

When simulated annealing is applied to a continuous optimized problem, neighborhood range adjustment is indispensable. This research proposed the NPSA method with a new neighborhood range generation mechanism that eliminates unnecessary parameters. The proposed method is found to be very effective for solving continuous optimization problems by SA.

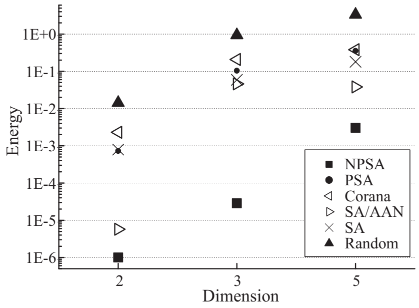


Fig. 2. energy value (Rastrigin function)

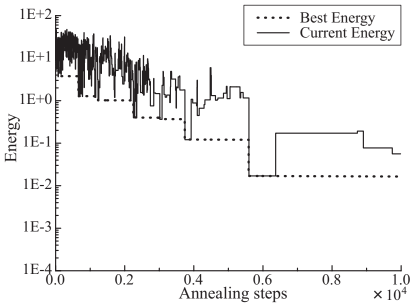


Fig. 5. energy history of PSA

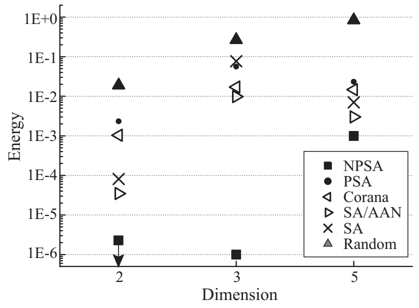


Fig. 3. Energy value (Griewank function)

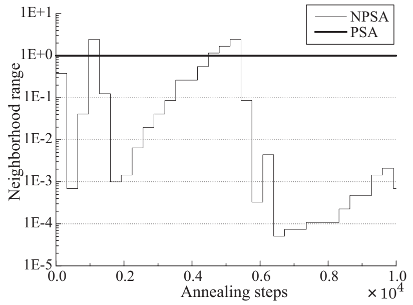


Fig. 6. history of neighborhood ranges

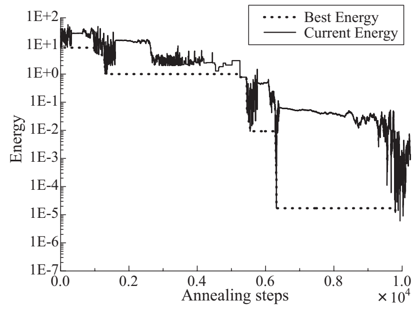


Fig. 4. energy history of NPSA

References

1. Hajime Kita (1997) Simulated Annealing. Japan Society for Fuzzy Theory and Intelligent Informatics, Vol. 9, No. 6, pp. 870-875
2. Aarts, E.(1989) Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.John Wiley&Sons
3. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953)Equation of State Calculation by Fast Computing Machines:Journ. of Chemical Physics, Vol. 21, pp.1087-1092
4. Ingber, L(1993) Simulated Annealing: Practice versus theory.Journal of Mathl.Comput.and Modelling.Vol.18, No.11, pp29-57
5. M.Kiku(1996)Invitation to genome information:Kyoritsu shuppan
6. Corana, A. (1987) Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm". ACM Trans. on Mathematical Software, Vol. 13, No. 3, pp. 262-280