Parallel Simulated Annealing using Genetic Crossover

Tomoyuki HIROYASU[†], Mitsunori MIKI[†], and Maki OGURA[††]

[†] Department of Knowledge Engineering,
[††] Graduate School of Engineering,
Doshisha University
Kyo-tanabe, Kyoto, 610-0321, JAPAN
Email: tomo@is.doshisha.ac.jp

Abstract

This paper proposes a new algorithm of a simulated annealing (SA): Parallel Simulated Annealing using Genetic Crossover (PSA/GAc). The proposed algorithm consists of several processes, and in each process SA is operated. The genetic crossover is used to exchange information between solutions at fixed intervals. While SA requires high computational costs, particularly in continuous problems, this operation reduces the computational cost. The proposed algorithm is applied to continuous test problems. It is found that the proposed algorithm can search the global optimum solution effectively, compared to distributed genetic algorithms and sequential simulated annealing.

keywords: Optimization Problems, Parallel Distributed Algorithms, Simulated Annealing, Genetic Crossover, Hybrid Algorithm

1 Introduction

Simulated Annealing (SA) is one of the emergent calculation algorithms that solves optimization problems, and is an effective technique for solving combination optimization problems[1]. SA is an algorithm that simulates the physical evolution of a solid from a high temperature state to a thermal equilibrium state[2]. SA searches randomly around the neighborhood of a present searching point. The next searching point can be accepted even when the fitness value of the next point is worse than that of the present. SA algorithms repeat these steps, and the optimization state is finally expected from given initial state[1][3]. Therefore, it can derive the global solution. However, SAs require huge computational cost. Specifically, SA takes much time finding the optimum solution in continuous problems. There are two solutions to this problem: performing SA in parallel and performing SA with other optimization algorithms.

Current research has proposed several types of parallel SA (PSA). One of the PSA models is Temperature Parallel Simulated Annealing (TPSA)[4]. In this algorithm, there are some processors and each processor has a process. Each process has a different fixed temperature. In each process, simple SA is performed and the information of the solutions is transferred after some steps.

An algorithm in which SA is utilized with other optimization algorithms is called a hybrid algorithm. Parallel Recombinative Simulated Annealing (PRSA)[5] and Thermodynamical Genetic Algorithm (TDGA)[6] are hybrid GAs. These algorithms use the concept of SA (Metropolis standard or the concept of temperature) at GA's "selection" process. However, there are few SA algorithms that use GA operations. Generally, it can be said that GA is good at searching globally and SA is good at searching locally. Therefore, the hybrid method of SA with GA operator is good at searching not only locally but also globally.

In this paper, we propose Parallel Simulated Annealing using Genetic Crossover (PSA/GAc). This algorithm is a hybrid SA using the GA operations. The proposed algorithm can reduce the computational cost even in continuous problems. The proposed algorithm was applied to some test functions and the effectiveness of the proposed algorithm will be discussed in this paper.

2 Parallel Simulated Annealing

2.1 Simulated Annealing

Simulated Annealing (SA) algorithm has basically three important processes: generation, acceptance criterion and cooling. SA searches a solution with one point. The process of generation creates the next searching point from the present point. The acceptance criterion then judges the transfer of the searching point from the present point to the generated point. This acceptance criterion consists of the temperature and the function value. Usually, the Metropolis standard is used as the acceptance criterion. The Metropolis standard is defined as follows;

$$A(x,x',T) = \begin{cases} 1 & if\ \Delta E \le 0 \\ \exp(-\frac{\Delta E}{T}) & otherwise \end{cases}$$

Because of this criterion, even when the value of the next point is worse than that of the present point, the next point can be accepted.

The operation of cooling decides the temperature in the next step[1]. Generally, the exponential annealing shown in Eq.1 is used.

$$T_{k+1} = \gamma T_k \quad (0.8 \le \gamma < 1) \qquad (1)$$

To derive the solution, the three operations- generation, acceptance criterion, and cooling- are iterated.

3 Parallel Simulated Annealing using Genetic Crossover

3.1 PSA/GAc

In this paper, we propose Parallel Simulated Annealing using Genetic Crossover (PSA/GAc). PSA/GAc is a hybrid method, and it uses parallel SA with the operations that are used in genetic algorithms. The flow of the concept is shown in Figure 1. In the proposed algorithm, there are plural processes and the sequential SA is operated in each process. After some steps, the crossover that is usually used in GAs is used to exchange the information between the solutions. We call this operation genetic crossover. We call a searching point an "individual", the total number of SA searching points the "population size" and annealing steps "number of generations". In optimization problems, the value of the objective function is made smaller or bigger. This value is the same as the value of the fitness in GAs and the value of the energy in SAs.

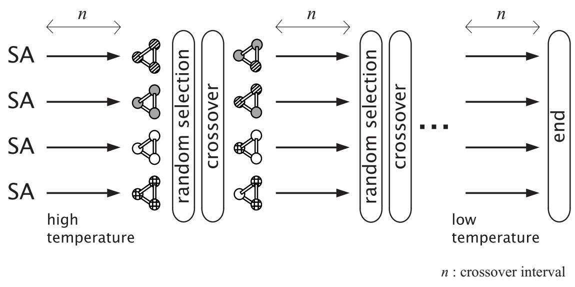

Figure 1: Simulated annealing using genetic crossover

In the genetic crossover, we randomly select two individuals as parents and generate two children by genetic crossover. The genetic crossover is shown in Figure 2. An individual consists of design variables and the design variables are real numbers. Therefore, the crossover is only performed between the variables. In the operation of genetic crossover, selection is also performed. After the crossover, there are two parents and two children. Then the two individuals that have higher evaluation values are selected. These two individuals become the next searching points. While SA requires high computational costs, particularly in continuous problems, this operation reduces the computational cost.

SAs in continuous problems need definitions of the neighborhood and terminal condition. The neighborhood and acceptance criteria are explained in sections

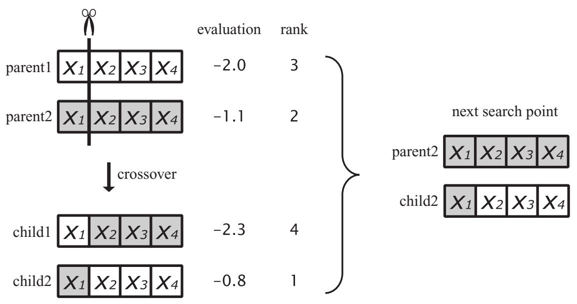

Figure 2: Crossover

3.2 and 3.3, respectively. In this paper, Metropolis criterion is used as the acceptance criterion and the exponential annealing $T_{k+1} = 0.93 \cdot T_k$, where k is the generation, is used as the cooling schedule.

3.2 Neighbourhood

In SAs, the neighborhood of the present searching point should be defined in order to find the next searching point. However, it is difficult to determine the neighborhood to search efficiently, especially in continuous problems. In this paper, Corana's algorithm is used to define the neighborhood. Corana proposes the SAs algorithm have an adjusting neighborhood range[7]. This algorithm adjusts the neighborhood range to keep the acceptance ratio to 50 % for an effective search.

3.3 Termination Condition

In SA, there are several methods for terminal conditions. SA algorithm accepts not only improvement solutions but also worse solutions. Therefore, when the annealing time is used as the terminal condition, the final solution cannot be the best solution in a simulation. Furthermore, it is very difficult to know the optimum annealing times. In this paper, we do not use an annealing time as a terminal condition, but the movement of solution as a terminal condition. It is possible to control quality of solutions. With this terminal condition, annealing does not stop in the middle of a search.

4 Numerical Examples

4.1 Comparison between PSA/GAc and SA using other GA operations

In the proposed algorithm, crossover is used for exchanging the information between individuals. Is this operation best for exchanging the information? To answer this question, other operations that can be used in SA and are compared to PSA/GAc. These operations are explained as follows;

- PSA using Elite Selection (elitePSA): The best individual that had the highest fitness value is copied to the other processes and these copies are used as the next searching points.

- PSA using Roulette Selection (roulettePSA): The next searching points are determined randomly by Roulette Selection. In Roulette Selection, the individual that has the higher value can be selected with high probability.

- PSA using Roulette Selection including Elite(e-roulettePSA): The next searching points are determined by Roulette Selection but the individual that has the highest fitness value will be one of the next points.

4.1.1 Test Functions

Rastrigin function (f_{Ra}) which has many local minima in global area and Griewank function (f_{Gr}) which has many local minima in a local area are used as continuous optimization test problems. These functions are shown as follows;

$$f_{Ra} = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)) \qquad (2)$$

$$f_{min} = 0, \ [x_i = 0], \quad n = 2$$

$$f_{Gr} = 1 + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}(\cos(\frac{x_i}{\sqrt{i}})) \qquad (3)$$

$$f_{min} = 0, \ [x_i = 0], \quad n = 2$$

Generally, GAs can find the optimum solution of the Rastrigin function easily. On the other hand, while SAs can find the optimum solution of the Griewank function easily, it is difficult to find the optimum solution with GAs.

In the following numerical examples, the simulation is terminated when the value of the solution does not change more than 1.0e-4 for 100 generations. Therefore, if the solution has a value smaller than 1.0e-4 , it can be said that a good solution is derived.

4.1.2 Parameter

The parameters used in the numerical examples are summarized in Table 1(value1). All results are the average of 10 trials. The initial temperature was chosen through preparative experiments.

Table 1: Parameter of parallel SAs

parameter	value 1	value 2
Population size	20, 200	400, 600
Initial temperature	5, 10, 20, 30	5, 10, 20, 30
Cooling rate	0.93	0.93
Communication interval	32	32
Neighborhood adjustment interval	8	8

4.1.3 Analysis of Results

Figure 3 is the result of the Rastrigin function with 20 population on each parallel SA (PSA). In Figure 3, the energy is equal to the value of the solution. PSA/GAc (crossoverPSA) always got a good solution and was not affected by the initial temperature. On the other hand, other three PSAs did not get good solutions.

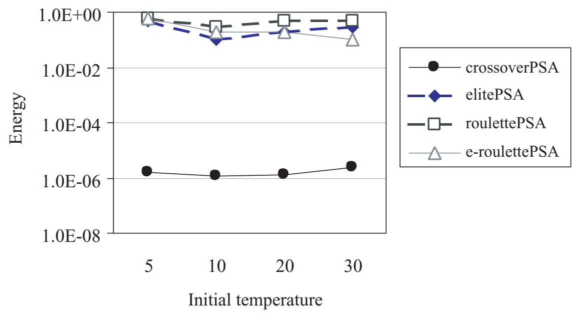

Figure 3: Results of Rastrigin function (population size 20)

In the case of 200 population, there are no differences between PSAs. All PSAs got good solutions.

Figure 4 and Figure 5 are the results of the Griewank function with 20 and 200 population, respectively. In Figure 4, it can be said that the results of PSA/GAc were slightly better than those of other PSAs. However, there were no significant differences between PSAs. In the case of 200 population that is shown in Figure 5, there are differences between PSAs. PSA using Roulette Selection and PSA using Roulette Selection including Elite never found a good solution and PSA using Elite Selection only found good solutions at the initial temperatures of 20 and 30. PSA/GAc (crossoverPSA) found good solutions with any initial temperature.

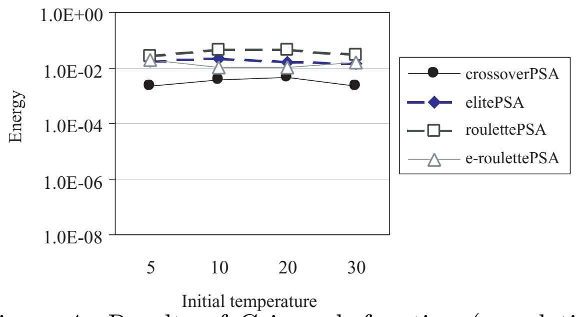

Figure 4: Results of Griewank function (population size 20)

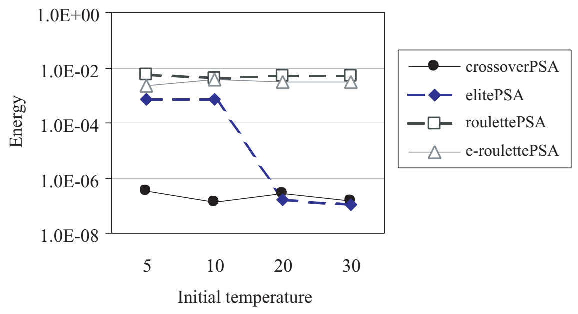

Figure 5: Results of Griewank function (population size 200)

The Griewank function has only one peak in the function landscape from a global view. However, it has many local minimums from a local view. Therefore, in the case of the small number of population, the searching point is trapped at the local minimum and the global optimum solution cannot be found. Because of this reason, the larger number of population needs to derive a better solution in the Griewank function.

These results show that among PSAs using GA operations, PSA using Genetic Crossover is the most effective. PSA using Elite Selection also has effective searching ability, but the possibility of reaching a good solution is lower than that of PSA/GAc. Therefore, it can be said that the searching ability of PSA using Elite Selection is lower than that of PSA/GAc.

4.2 Searching ability of PSA/GAc

In the numerical examples in Section4.1, we found that Parallel Simulated Annealing using Genetic Crossover is more effective, compared to PSAs with other operations. In this section, PSA/GAc is compared to the Distributed Genetic Algorithm (DGA) and Sequential Simulated Annealing (SSA) through numerical examples.

4.2.1 Test Functions

Rastrigin function, Griewank function and Rosenbrock function are used as continuous optimization test problems. These functions are explained as follows;

$$f_{Ra} = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)) \qquad (4)$$

$$f_{min} = 0,\ [x_i = 0],\quad n = 10, 30$$

$$f_{Gr} = 1 + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}(\cos(\frac{x_i}{\sqrt{i}})) \qquad (5)$$

$$f_{min} = 0,\ [x_i = 0],\quad n = 10, 30$$

$$f_{Ro} = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \qquad (6)$$

$$f_{min} = 0,\ [x_i = 1],\quad n = 10, 30$$

The dimensions are equal to 10 and 30. We retain the definition of a good solution from Section 4.1.1.

4.2.2 Parameter

The parameters are summarized in Table 1(value2). The results that are shown in the following sections were the average of 10 trials. The initial temperatures were derived from preliminary experiments.

4.2.3 Searching ability of PSA/GAc

In Figure 6, the results of a 10 dimension Rastrigin function with PSA/GAc are shown. In the figure, (a) is the case of 400 population, and (b) is the case of 600 population. 'Average' means the average of 10 trials, 'best' means the best solution of 10 trials, and 'average(good solutions)' means the average of the good results. The values of bar graph express the number of good solutions in 10 trials.

Figure 6 shows that PSA/GAc can reach the good solution with high probability, even if the problem has 10 dimensions. In the case of a 400 population, figure (a) shows that the good solutions were derived with any initial temperature and an adequate search was performed.

The results of the 30 dimension Rastrigin function problem are shown in Figure 7. Figure 7 shows that good solutions were not derived in the case of a 30 dimension Rastrigin function. SAs are not good at finding solutions in a problem, such as the Rastrigin function, whose function landscape has many local optima in a global view. By comparing figures (a) and (b), it becomes apparent that the probability of finding a good solution to a problem with a large population size is higher than the probability of finding a solution to a problem with a small population size.

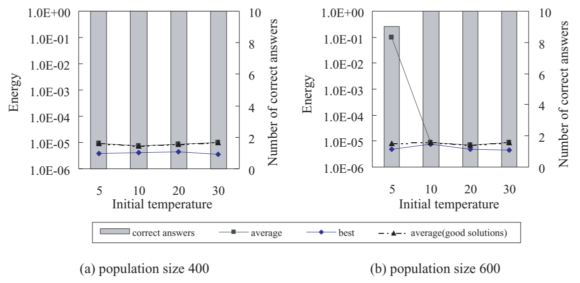

Figure 6: Results of 10 dimension Rastrigin function

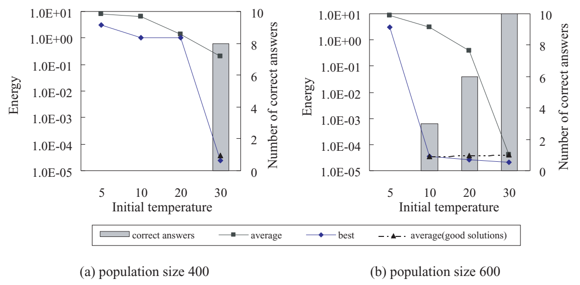

Figure 7: Results of 30 dimension Rastrigin function

Figure 8 shows the results of a 10 dimension Griewank function. In this case, good solutions were derived at any initial temperature.

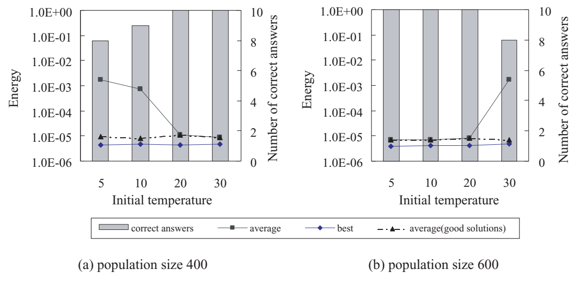

Figure 8: Results of 10 dimension Griewank function

Figure 9 shows the results of a 30 dimension Griewank function. This figure indicates that all results were better than the results of the 10 dimension of Griewank. The Griewank function is expressed by Eq.5, and the third term of this function is the product of each dimension. Therefore, when there are many dimensions, the first term's affect upon the value of the object function is stronger than the other terms. For this reason, a Griewank function is easy to solve if it has many dimensions.

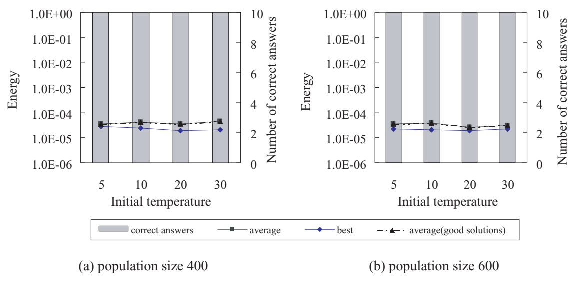

Figure 9: Results of 30 dimension Griewank function

Figure 10 is the result of a 10 dimension Rosenbrock function, and Figure 11 is the result of a 30 dimension Rosenbrock function. GAs are not good at finding the optimum solution to Rosenbrock functions. However, in this numerical example, PSA/GAc always derived good solutions.

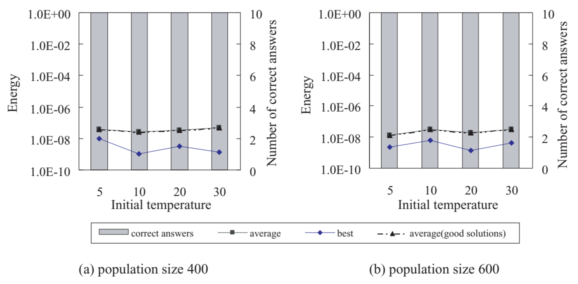

Figure 10: Results of 10 dimension Rosenbrock function

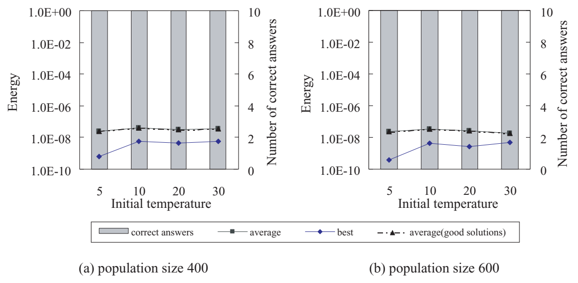

Figure 11: Results of 30 dimension Rosenbrock function

4.2.4 Comparison between PSA/GAc and Sequential SA

In order to verify the searching ability of PSA/GAc, we compared PSA/GAc and Sequential Simulated Annealing (SSA). In PSA/GAc, the solution was derived with 8000 generations with 400 population. Then, we define the SSA-long compute SSA for 3200000 (8000 generations * 400 population). We define SSA-short to compute SSA for 8000 generations for 400 times. The results were compared to those of PSA/GAc. The initial temperatures were 10 in each algorithm. All results are the average of 10 trials. Table 2 is the results of Rastrigin function. In the case of 10 dimensions, PSA/GAc reached a good solution in all 10 trials. On the other hand, the two types of SSAs did not reach good solutions at all. In the case of 30 dimensions, both PGA/GAc and SSA did not reach good solutions. However, the solution values of PGA/GAc were better than those of the SSA.

Table 2: Comparison between PSA/GAc and sequential SA (Rastrigin function)

	PSA	SSA-long	SSA-short
10 dimensions			
Solution	7.64e-6	30.0	25.2
Success rate	1.0	0.0	0.0
30 dimensions			
Solution	6.67	226	216
Success rate	0.0	0.0	0.0

Table 3 shows the results of the Griewank function. The Sequential SA did not reach a good solution in both the 10 dimension and the 30 dimension Rastrigin functions. However, PSA/GAc reached the good solution with high probability.

Table 3: Comparison between PSA/GAc and sequential SA (Griewank function)

	PSA	SSA-long	SSA-short
10 dimensions			
Solution	7.49e-4	3.78	0.273
Success rate	0.9	0.0	0.0
30 dimensions			
Solution	4.13e-5	0.459	0.592
Success rate	1.0	0.0	0.0

Table 4 is the result of the Rosenbrock function. It is obvious that the results of PSA/GAc are superior to those of simple SA. This result is also derived from the results of the Griewank function test. From Table 3 and Table 4, we can find that SSA cannot find good solutions, even when the annealing time is large. It is clear that PSA/GAc is more effective than SSA.

Table 4: Comparison between PSA/GAc and sequential SA (Rosenbrock function)

	PSA	SSA-long	SSA-short
10 dimensions			
Solution	2.60e-8	1.92e-3	2.09e-3
Success rate	1.0	0.0	0.1
30 dimensions			
Solution	4.03e-8	1.48e-3	2.59e-3
Success rate	1.0	0.0	0.0

4.2.5 Comparison between PSA/GAc and Distributed GA

In this Section, PSA/GAc is compared to the distributed genetic algorithm (DGA) and dual individual GA (DuGA). The population size, the number of bits and the terminal condition were set up as follows: Population size of PSA/GAc was 400, and the population of DGA and DuGA was 400 (20 individuals * 20 islands) and 400(2 individuals *200 islands), respectively. The simulation of PSA/GAc was terminated within 8000 generations and the DGA and DuGA were also terminated within 8000 generations. In each algorithm, if the optimum solution was found, the simulation was terminated before 8000 generations. The bit string of GA was 30 per dimension. Table 5, Table 6, and Table 7 are the results of the Rastrigin function, Griewank function, and Rosenbrock function, respectively. The initial temperature of PSA was 10. All results are the average of 10 trials.

Table 5: Comparison between PSA/GAc and GA (Rastrigin function)

	PSA	GA(20*20)	duGa
Optimum	1.0e-5$order$	0	0
10 dimensions			
Success rate	1.0	1.0	1.0
Evaluations	3034881	773940	457000
30 dimensions			
Success rate	0.0	0.1	0.2
Evaluations	3117641	3181940	3121600

From Table 6 and Table 7, it is clear that PSA/GAc can reach a good solution in the high probability even when the applied test functions pose difficult problems for GAs. Therefore, PSA/SAc is suitable for problems that can be solved by SA, but pose difficulties for GA. However, Table 5 shows that DGA is superior to the proposal algorithm in the problems where the GA is good at finding the solution.

Table 6: Comparison between PSA/GAc and GA (Griewank function)

	PSA	GA(20*20)	duGa
Optimum	1.0e-5$order$	0	0
10 dimensions			
Success rate	0.9	0.0	0.2
Evaluations	3008201	3200400	2676960
30 dimensions			
Success rate	1.0	0.7	0.9
Evaluations	3118041	2922120	1819760

Table 7: Comparison between PSA/GAc and GA (Rosenbrock function)

	PSA	GA(20*20)	duGa
Optimum	1.0e-8$order$	0	0
10 dimensions			
Success rate	1.0	0.0	0.0
Evaluations	2750721	3200400	3200400
30 dimensions			
Success rate	1.0	0.0	0.0
Evaluations	2723441	3200400	3200400

5 Conclusions

In this paper, we proposed a new algorithm of SA: Parallel Simulated Annealing using Genetic Crossover (PSA/GAc). In the proposed algorithm, there are several simple SA processes and each process is operated parallel. After some steps, the information of the searching point is transferred among the processes. This information transformation is performed by crossover that is usually used in genetic algorithms.

Apart from the genetic crossover, there are other operations that are used in genetic algorithms for transforming the information. Those are elite selection, roulette selection, and so on. The candidate operations were applied to the test functions and the results were compared with those of the proposed algorithm. The result showed that the genetic crossover is better than the other candidate operations. Therefore, it can be concluded that the genetic crossover is effective operation for transforming information in parallel SAs. Through the numerical examples, the proposed algorithm was also compared to other emergent algorithms, sequential SAs and distributed GAs. It was found that the results of the proposed algorithm are superior to those of sequential SAs. It was also found that the proposed algorithm is good at finding solutions in problems that are not suited to GAs. Generally, SAs are algorithms used for finding the solutions of discrete problems. However PSA using Genetic Crossover is effective for a variety of continuous problems.

References

[1] Bruce E. Rosen and Ryohei Nakano. Simulated annealing -basics and recent topics on simulated annealing-(in japanese). *Proceeding of Japanese Society for Artificial Intelligence*, Vol. 9, No. 3, 1994.

[2] E. H. L. Aarts and J. H. M. Korst. *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons, 1989.

[3] Hajime Kita. Simulated annealing (in japanese). *Proceeding of Japan Society for Fuzzy Theory and Systems*, Vol. 9, No. 6, 1997.

[4] Kenso Konishi, Kazuo Taki, and Kouichi Kimura. Temperature parallel simulated annealing algorithm and its evaluation. *Transaction of Information Processing Society of Japan*, Vol. 36, No. 4, pp. 797–807, 4 1995.

[5] S. W. Mahfoud and D. E. Goldberg. A genetic algorithm for parallel simulated annealing. *Parallel Problem Solving from Nature 2*, pp. 301–310, 1992.

[6] Naoki Mori, Junji Yoshida, and Hajime Kita. Suggestion of thermodynamical selection rule in genetic algorithm (in japanese). *Transaction of Institute of Systems, Control and Information Engineers*, Vol. 9, No. 2, pp. 82–90, 1996.

[7] A. Corana, M. Marhesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the 'simulated annealing' algorithm. *ACM Trans. on Mathematical Sortware*, Vol. 13, No. 3, pp. 262–280, 1978.

Proceedings of the IASTED International Conference PARALLEL AND DISTRIBUTED COMPUTING AND SYSMTEMS November 6-9, 2000, Las Vegas, Nevada USA pp. 139-144