Computer Assisted Mechanics and Engineering Sciences, 15: 85–98, 2008.

Improvement of evolutionary algorithm based on schema exploiter

Takashi Maruyama, Eisuke Kita
Graduated School of Information Sciences, Nagoya University, Nagoya 464-8601, Japan

(Received October 9, 2006)

Stochastic Schemata Exploiter (SSE) is one of the evolutionary optimization algorithms for solving the combinatorial optimization problems. We present the Extended SSE (ESSE) algorithm which is composed of the original SSE and new ESSE operations. The ESSE is compared with the original SSE, simple genetic algorithm (SGA), and GA with Minimal Generation Gap (MGG) in some test problems in order to discuss its features.

Keywords: Genetic Algorithm (GA), Stochastic Schemata Exploiter (SSE), Extended SSE (ESSE), Minimal Generation Gap (MGG)

1. Introduction

In most of the combinational optimization problems, the objective function spaces have so-called "big valley structure" [1]. In the problems with big valley structure, there is often a real (global) optimum solution near quasi-optimal solutions. The evolutionary algorithms are considered to be effective for solving such optimization problems [2–4].

Genetic algorithm (GA) has been firstly presented by J. Holland in 1975 [2]. The GA, which is the algorithm to mimic the natural evolution, is widely applied to optimization, adaptation and learning problems. The basic algorithm of the GA is often called as simple genetic algorithm (SGA) [3]. Many improved algorithms are derived from the SGA. The search performance of the SGA can be discussed from the viewpoints of the early convergence and the evolutionary stagnation [4, 5]. The early convergence means that all individuals are rapidly attracted to a local optimum solution and therefore, the global optimum solution cannot be found. The evolutionary stagnation means that the convergence speed becomes slower as the iterative process goes. Once a quasi-optimal solution is found, it is generally difficult for the SGA to find better ones. For overcoming these difficulties, Sato *et al.* has presented new generational alternation model named as Minimal Generation Gap (MGG) [6]. The application of the GA model with MGG to several actual problems reveals that MGG is very effective for such actual optimization problems [7].

Stochastic Schemata Exploiter (SSE) has been presented by Aizawa in 1994 [8]. In the traditional SGA, better individuals are selected from a population and new individuals are generated from them by applying genetic operators such as crossover and mutation. The SSE algorithm is very different from the SGA algorithm. In the SSE, the sub-populations are determined according to the semi-order relation of the sub-populations from the descending order of the fitness of the individuals. Common schemata are extracted from the individuals in the sub-populations and new individuals are generated from them. Selection and crossover operations are not necessary in SSE. Since the SSE algorithm tends to search better solutions near good solutions which are already found, it is adequate for solving actual optimization problems with function spaces of big valley structure. The SSE has two attractive features. Firstly, the variety of control parameters is smaller than the GA. Since the selection and crossover operations are not necessary in SSE, control parameters are population size and mutation rate alone. Secondly, the convergence speed is very fast. Instead, in the

SSE, all individuals often converges to the local optimal solution. For overcoming this difficulty, this paper describes Extended Stochastic Schemata Exploiter (ESSE). In the ESSE, after the common schemata are extracted from the individuals in the sub-populations in order to define the common schemata list, the schemata list is modified by deleting same schemata from the list, updating similar schemata and so on. The additional operations enhance the diversity of the population and therefore, the search performance of ESSE can be improved fairly.

The remaining of the paper is organize as follows. In Sections 2 and 3, the algorithms of the SSE and ESSE are explained. In Section 4, their performance is compared in some numerical examples. Finally, the results are concluded in Section 5.

2. Stochastic Schemata Exploiter

2.1. SSE algorithm

We will show the algorithm of the stochastic schemata exploiter (SSE). Besides, the SSE algorithm is illustrated in Fig. 1.

1. *Initial population generation*
   An initial population is defined with randomly generating M individuals.

2. *Fitness function estimation*
   The fitness functions of individuals are evaluated, and the individuals are ranked according to the descending order of their fitness.

3. *Convergence criterion*
   If the criterion is satisfied, the process stops.

4. *Sub-population definition*
   M sub-populations are defined according to the semi-order relation from the fitness function order of the individuals.

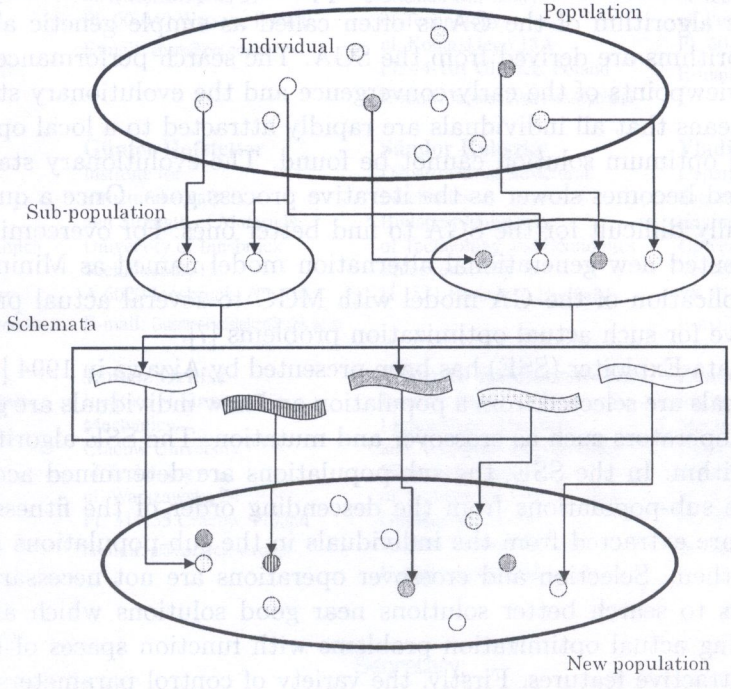

Fig. 1. Stochastic schemata exploiter (SSE)

5. *Common schemata extraction*
   Common schemata are extracted from the individuals in the sub-populations.
6. *New population generation*
   New individuals are generated from the schemata in order to define a new population.
7. *Time step incrementation*
   A time step is incremented and the process returns to step 2.

The particular processes in the SSE are defining sub-populations, extracting common schemata, and generating new individuals. So, we would like to explain them in the followings.

2.2. Defining sub-populations

The sub-populations are generated according to the semi-order relation between the sub-populations. In the followings, we will explain the semi-order relation and then, how to define sub-populations.

2.2.1. Semi-order relation

The population P is composed of the individuals c_1, c_2, ... c_M, which are numbered according to the descending order of their fitness. Therefore, the individual c_k denotes the k-th best individuals in the population P. The symbol S denotes the sub-population of the population P. When the individual c_k is excluded from S, a new population is represented as $S - c_k$. The operator $\cup$ denotes the union of sets.

When the worst individual in the sub-population S is numbered as $L(S)$, the following semi-order relation is held in the sub-populations of the population P.

1. Since the individual $c_{(L(S))}$ is the worst one in the sub-population S, the individual $c_{(L(S)+1)}$ is worse by one order than the individual $c_{(L(S))}$. When the individual $c_{(L(S)+1)}$ is added to a sub-population S, the new sub-population is defined as $S \cup c_{(L(S)+1)}$. A first semi-order relation is given as

$$f(S) \geq f\left(S \cup c_{(L(S)+1)}\right) \tag{1}$$

   where $f(S)$ denotes the average fitness of the individuals in the sub-population S.

2. When the individual $c_{(L(S))}$ is replaced with the individual $c_{(L(S)+1)}$, the new population is defined as $(S - c_{L(S)}) \cup c_{(L(S)+1)}$. A second semi-order relation is given as

$$f(S) \geq f\left(\left(S - c_{L(S)}\right) \cup c_{(L(S)+1)}\right). \tag{2}$$

2.2.2. Sub-population

The use of semi-order relation gives the following order of sub-populations. Since it is obvious that the best sub-population is composed of the best individual c_1 alone, we have a first sub-population

$$S_1 = \{c_1\}.$$

When adding the individual c_2 to the sub-population $S_1 = \{c_1\}$ according to the semi-order relation (1), we have a second sub-population

$$S_2 = \{c_1, c_2\}.$$

When replacing the individual c_1 in the sub-population S_1 with the individual c_2 according to the semi-order relation (2), we have a third sub-population

$$S_3 = \{c_2\}.$$

When adding the individual c_3 to the sub-population $S_2 = \{c_1, c_2\}$ according to the semi-order relation (1), we have a fourth sub-population

$$S_4 = \{c_1, c_2, c_3\}.$$

When replacing the individual c_2 in the sub-population S_2 with the individual c_3 according to the semi-order relation (2), we have

$$S_4 = \{c_1, c_3\}.$$

We can define the other sub-populations in the similar way.

2.3. Extracting common schemata

After defining the sub-populations, the common schemata are extracted as the product of the chromosome of the individuals in the sub-populations.

2.4. Generating new individuals

The extracted schemata are composed of three characters; "0", "1", and "$*$". So, the new individuals are defined by randomly replacing "$*$" by "0" or "1".

3. Extended stochastic schemata exploiter

3.1. Basic concept

The SSE algorithm often generates identical or very similar schemata from different sub-populations. If the identical schemata are deleted from the common schemata list, the diversification of the new population can be improved. This is the basic idea of the extended SSE (ESSE) algorithm.

Four relations are held among the schemata in the schemata list:

Case 1: two schemata are identical,

Case 2: one Schemata is included into the other one,

Case 3: they are partially identical, and

Case 4: they are different.

For the cases 1, 2 and 3, the following ESSE operations are applied to the schemata list.

In the following sentences, the symbol A and B denote the schemata extracted from the sub-population S_A and S_B, respectively, and the average fitness of the individuals in the sub-population S_A is defined as $f(S_A)$.

3.1.1. ESSE operation 1

If the schemata A and B are identical, the following processes are performed.

1. A is kept and B is deleted from the schemata list.

2. A common schema is extracted from $S_A \cup S_B$.

3.1.2. ESSE operation 2

If the schema A is included into the schema B, it is considered that the schema B is grown up from the schema A. In this case, the following process is performed.

1. If $f(S_A) > f(S_B)$, A is kept and the common schema is extracted from $S_A \cup S_B$.

2. If $f(S_A) \leq f(S_B)$, B is kept and the common schema is extracted from $S_A \cup S_B$.

3.1.3. ESSE operation 3

If the schema A and B are partially identical, the following processes are performed.

1. If $f(S_A) \geq f(S_B)$, A is kept. If not so, B is kept.

2. A common schema is extracted from $S_A \cup S_B$.

3.2. ESSE algorithm

The ESSE algorithm is composed of the original SSE algorithm and one or more of the above ESSE operations. The algorithm of the ESSE is illustrated in Fig. 2. The process in summarized as follows.

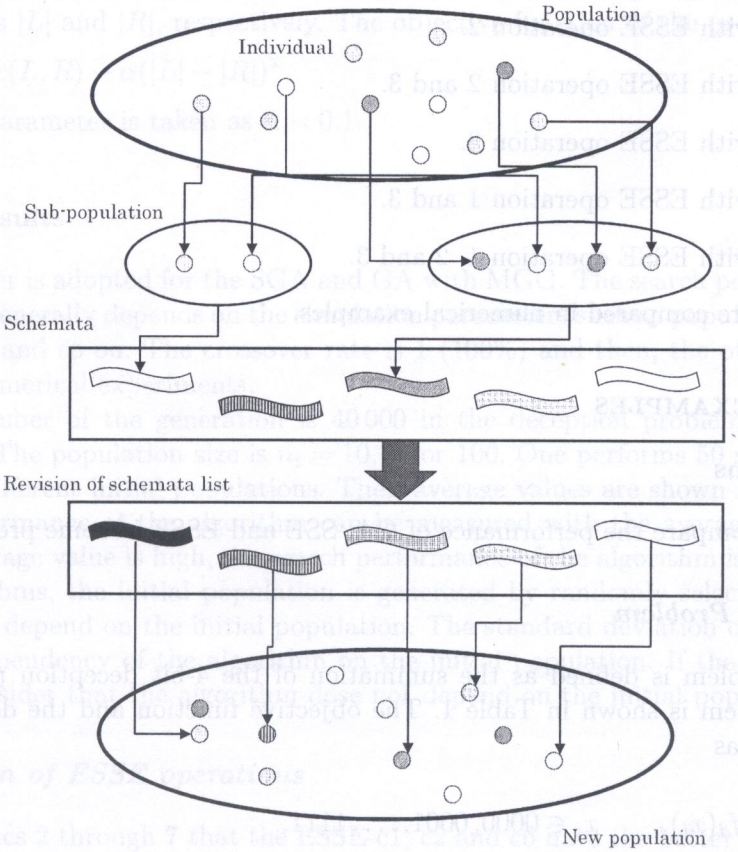

Fig. 2. Extended stochastic schemata exploiter (ESSE)

1. Initial population generation
2. Fitness function estimation
3. Convergence check
4. Sub-population generation
5. Common schemata extraction
6. ESSE operations
7. New individuals generation
8. Time step incrementation

The process 6 is added to the original SSE. After the common schemata are extracted in the step 5, ESSE operations reconstruct the list of the common schemata in the step 6.

3.3. ESSE Family

The ESSE is composed of the original SSE and one or more ESSE operations. So, we can define the seven ESSE algorithms according to the selection of ESSE operations. They are named as c_1 , c_2 , $\cdots$, c_7 .

c1: Original SSE with ESSE operation 1.

c2: Original SSE with ESSE operation 1 and 2.

c3: Original SSE with ESSE operation 2.

c4: Original SSE with ESSE operation 2 and 3.

c5: Original SSE with ESSE operation 3.

c6: Original SSE with ESSE operation 1 and 3.

c7: Original SSE with ESSE operation 1, 2 and 3.

Their features are compared in numerical examples.

4. Numerical examples

4.1. Test problems

We would like to compare the performance of GA, SSE and ESSE in some problems [9].

4.1.1. Deception Problem

The deception problem is defined as the summation of the 4-bit deception problems [9]. The 4-bit deception problem is shown in Table 1. The objective function and the design variable of the problem is defined as

$$f_{deception} = \sum_{i=1}^{n} f_d(x_i), \qquad x_i \in 0000, 0001, \ldots, 1111, \tag{3}$$

where n denotes the number of 4-bit deception problem and $n = 10$.

Table 1. Part solutions of deceptive problem

$f_d(1111) = 30$	$f_d(0000) = 28$	$f_d(0001) = 26$	$f_d(0010) = 24$
$f_d(0100) = 22$	$f_d(1000) = 20$	$f_d(0011) = 18$	$f_d(0101) = 16$
$f_d(0110) = 14$	$f_d(1001) = 12$	$f_d(1010) = 10$	$f_d(1100) = 8$
$f_d(1110) = 6$	$f_d(1101) = 4$	$f_d(1011) = 2$	$f_d(0111) = 0$

4.1.2. Knapsack problem

When there are n baggages in a knapsack, the knapsack problem is defined as the maximization of the value of the knapsack without exceeding the weight limit b. The problem is defined as

$$\max_{\{x_i\}} \sum_{i=1}^{n} c_i x_i \qquad \text{subject to} \quad \sum_{i=1}^{n} a_i x_i \leq b, \qquad x_i \in 0, 1\,, \tag{4}$$

where the weight and the value of the baggage i are referred to as a_i and c_i, which are randomly taken within $1 \leq a_i\,,\ c_i \leq 100$. Besides, $b = 10000$ and $n = 400$.

4.1.3. Graph partitioning problem

Given a graph G with the set of vertex V and the set of edges E that determines the connectivity between the nodes. The graph partitioning problem consists on dividing G into k disjoint partitions. The object is to minimize k and reduce the imbalance of the weight of the sub-domains. In this paper, the graph problem G124.08 [10] is adopted.

When the graph G is divided into two sets L and R, the total numbers of vertex included in the sets are expressed as $|L|$ and $|R|$, respectively. The objective function of the problem is defined as

$$f_{graph}(L, R) = -c(L, R) - \alpha(|L| - |R|)^2 \tag{5}$$

where the penalty parameter is taken as $\alpha = 0.1$.

4.2. Numerical results

A two-point crossover is adopted for the SGA and GA with MGG. The search performance of evolutionary algorithms generally depends on the simulation parameters such as population size, crossover rate, mutation rate and so on. The crossover rate is 1 (100%) and then, the other parameters are determined from numerical experiments.

A maximum number of the generation is 40 000 in the deception problem and 10 000 in the knapsack problem. The population size is $n_i = 10, 50$ or 100. One performs 50 simulations for each problem from the different initial populations. Their average values are shown in figures.

The search performance of the algorithm can be measured with the average value of the final solutions. If the average value is high, the search performance of the algorithm is good. Since, in the evolutionary algorithms, the initial population is generated by randomly selected individuals, the final solutions often depend on the initial population. The standard deviation of the final solutions can measure the dependency of the algorithm on the initial population. If the standard deviation is small, we can consider that the algorithm dose not depend on the initial population very well.

4.2.1. Comparison of ESSE operations

We notice from Tables 2 through 7 that the ESSE-c1, c2 and c6 have the better search performance than the others. Since ESSE-c1, c2 and c6 have the ESSE operation 1, we can imagine that ESSE operation 1 is specially effective for improving the SSE.

Table 2. Average values of final solutions (Deception problem)

N_i	10	50	100
c1	298.24	300.00	300.00
c2	298.92	300.00	300.00
c3	297.32	300.00	300.00
c4	297.84	299.56	299.12
c5	298.12	300.00	300.00
c6	297.64	300.00	300.00
c7	291.40	290.52	290.80

Table 3. Standard deviation of final solutions (Deception problem)

N_i	10	50	100
c1	2.81	0.00	0.00
c2	2.30	0.00	0.00
c3	2.73	0.00	0.00
c4	2.11	0.82	1.21
c5	2.81	0.00	0.00
c6	3.11	0.00	0.00
c7	2.00	2.18	2.33

Table 4. Average values of final solutions (Knapsack problem)

N_i	10	50	100
c1	16107.1	16150.2	16153.5
c2	16094.2	16149.5	16154.0
c3	16066.3	16135.7	16134.3
c4	16074.4	16121.3	16128.5
c5	16090.9	16142.9	16147.7
c6	16103.5	16148.1	16153.7
c7	15738.6	15644.9	15663.4

Table 5. Standard deviation of final solutions (Knapsack problem)

N_i	10	50	100
c1	13.5	2.7	1.4
c2	14.6	3.0	1.0
c3	17.5	7.8	7.6
c4	19.8	10.6	7.0
c5	17.8	4.8	3.7
c6	15.0	4.6	1.6
c7	82.4	79.1	82.7

Table 6. Average values of final solutions (Graph partitioning problem)

N_i	10	50	100
c1	−186.24	−183.3	−182.78
c2	−186.3	−183.22	−182.75
c3	−187.52	−184.6	−183.36
c4	−187.71	−184.02	−183.21
c5	−188.	−185.06	−184.69
c6	−185.98	−183.02	−183.42
c7	−188.27	−190.1	−189.63

Table 7. Standard deviation of final solutions (Graph partitioning problem)

N_i	10	50	100
c1	4.88	3.59	3.11
c2	5.29	3.47	2.95
c3	4.97	3.66	3.73
c4	4.64	3.33	3.34
c5	5.67	4.09	4.05
c6	4.90	4.25	3.69
c7	5.24	6.03	6.04

When focusing the ESSE-c2 which has ESSE operation 1 and 2, we notice that it has good search performance at the deception problem with $n_i = 10$ and the knapsack problem with $n_i = 100$.

Finally, we can conclude as follows.

- The ESSE operation 1 is most effective for improving the SSE.
- The ESSE operation 2 and 3 is less effective than the ESSE operation 1. In some cases, however, they can enhance the performance of the ESSE operation 1.

4.2.2. Comparison of ESSE, SSE, SGA, and GA with MGG

In the previous section, we noticed that the ESSE-c1 was the best among the ESSE algorithms. So, we will compare the ESSE-c1 with the SSE, SGA and GA with MGG.

Deception problem

Convergence history of the best individual at every generation is shown in Fig. 3. The history of the standard deviation is in Fig. 4. The abscissa and the ordinate denote the generation and the

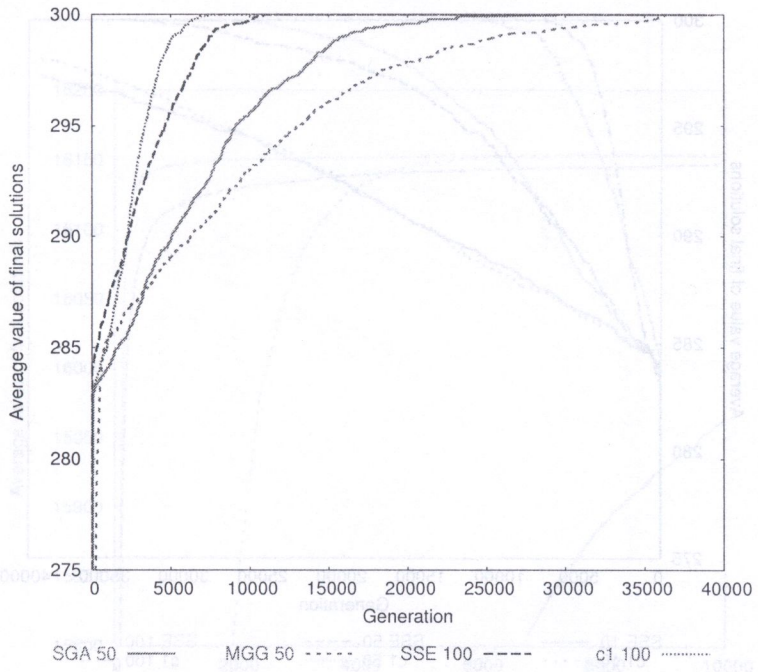

Fig. 3. Average values of solutions of SGA, MGG, SSE and c1 on deception problem

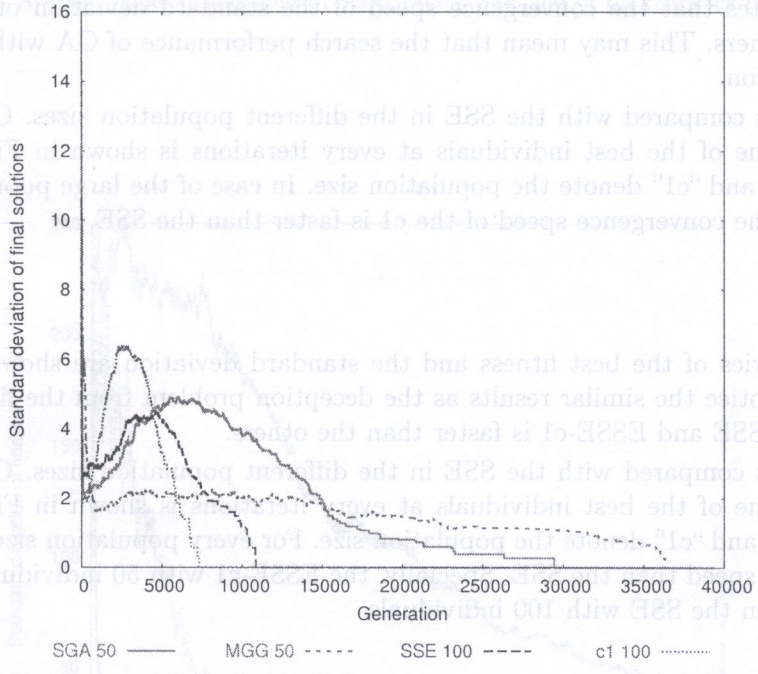

Fig. 4. Standard deviation of solutions of SGA, MGG, SSE and c1 on deception problem

best fitness and the standard deviation, respectively. The number following the algorithm name denotes the population size n_i. The population size is determined from numerical experiments for each algorithm. In this case, the population size for SGA and GA with MGG is 50, which is smaller than that for SSE and ESSE-c1. The numerical experiments indicated that 50 is more adequate than 100 for the population size of the SSE and ESSE-c1. So, we adopted it. We notice from Fig. 3 that the convergence speed of SSE and ESSE-c1 are faster than the others and that the ESSE-c1 is the fastest among them.

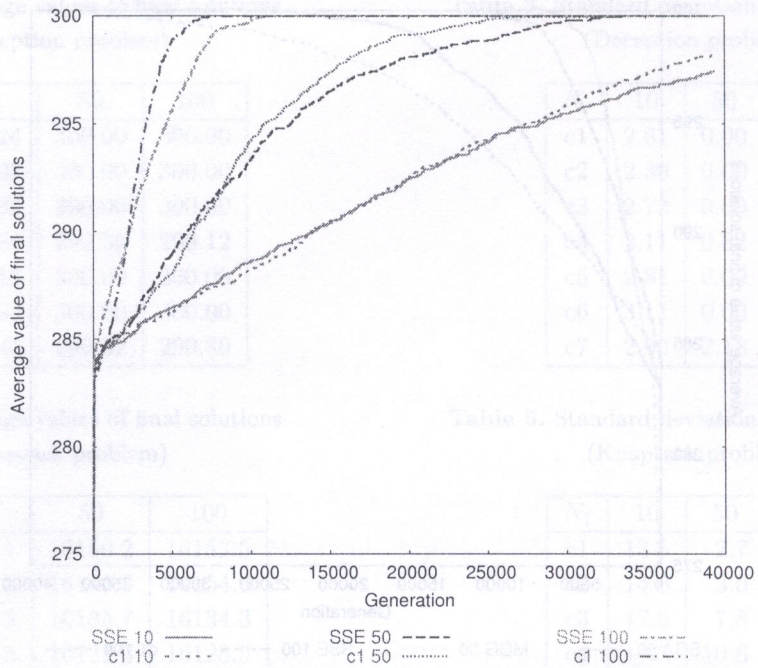

Fig. 5. Average values of solutions of SSE and c1 on deception problem

Figure 4 indicates that the convergence speed of the standard deviation of GA with MGG is slower than the others. This may mean that the search performance of GA with MGG depends on the initial population.

The ESSE-c1 is compared with the SSE in the different population sizes. Convergence history of the average value of the best individuals at every iterations is shown in Fig. 5. The numbers following to "SSE" and "c1" denote the population size. In case of the large population size (50 and 100 individuals), the convergence speed of the c1 is faster than the SSE.

Knapsack problem

Convergence histories of the best fitness and the standard deviation are shown in Figs. 6 and 7, respectively. We notice the similar results as the deception problem from the figures; i.e., the convergence speed of SSE and ESSE-c1 is faster than the others.

The ESSE-c1 is compared with the SSE in the different population sizes. Convergence history of the average value of the best individuals at every iterations is shown in Fig. 8. The numbers following to "SSE" and "c1" denote the population size. For every population sizes, the ESSE-c1 has faster convergence speed than the SSE. Specially, the ESSE-c1 with 50 individuals can find slightly better solution than the SSE with 100 individuals.

Graph partitioning problem

Convergence histories of the best fitness and the standard deviation are shown in Figs. 9 and 10, respectively. We notice that the convergence speed of the SSE and the ESSE-c1 is much faster than the MGG and that the ESSE-c1 can find the best solution among them. As shown in Fig. 10, the SSE and the ESSE-c1 has faster convergence speed than the others.

The ESSE-c1 is compared with the SSE in the different population sizes. Convergence history of the average value of the best individuals at every iterations is shown in Fig.11. We can recognize that the ESSE-c1 with 50 individuals can find much better solution than the SSE with 100 individuals although the population size is one-half.

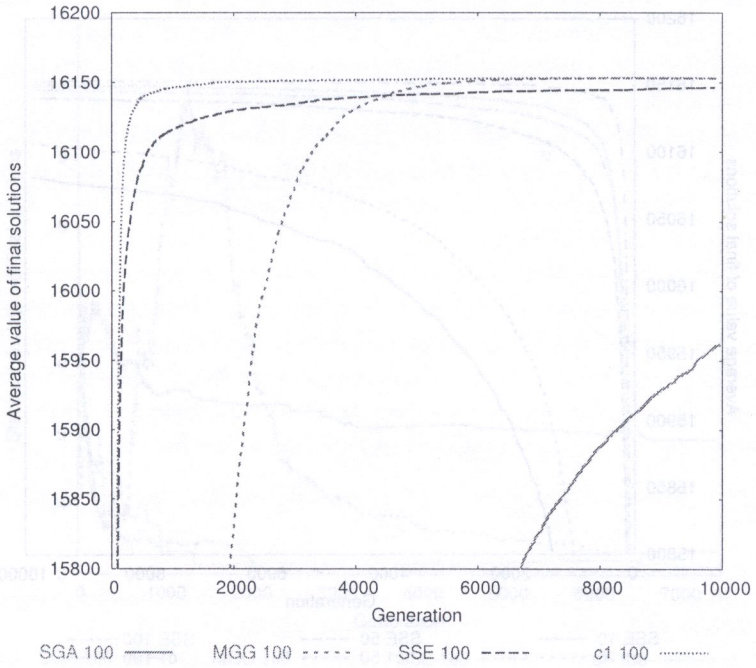

Fig. 6. Average values of solutions of SGA, MGG, SSE and c1 on knapsack problem

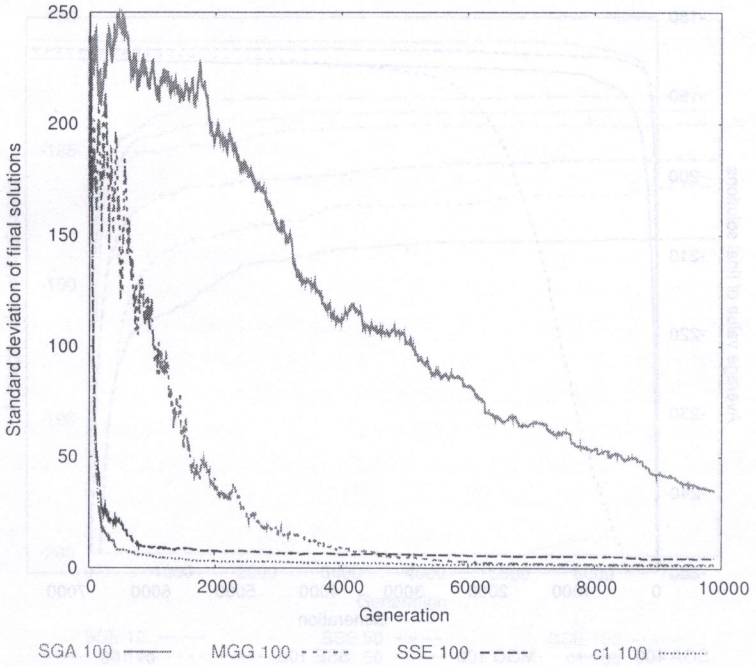

Fig. 7. Standard deviation of solutions of SGA, MGG, SSE and c1 knapsack problem

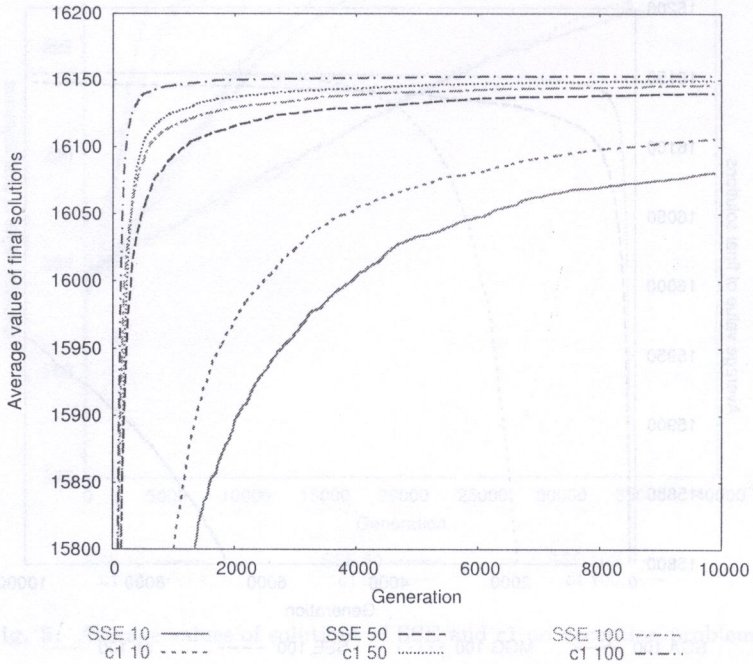

Fig. 8. Average values of solutions of SSE and c1 on knapsack problem

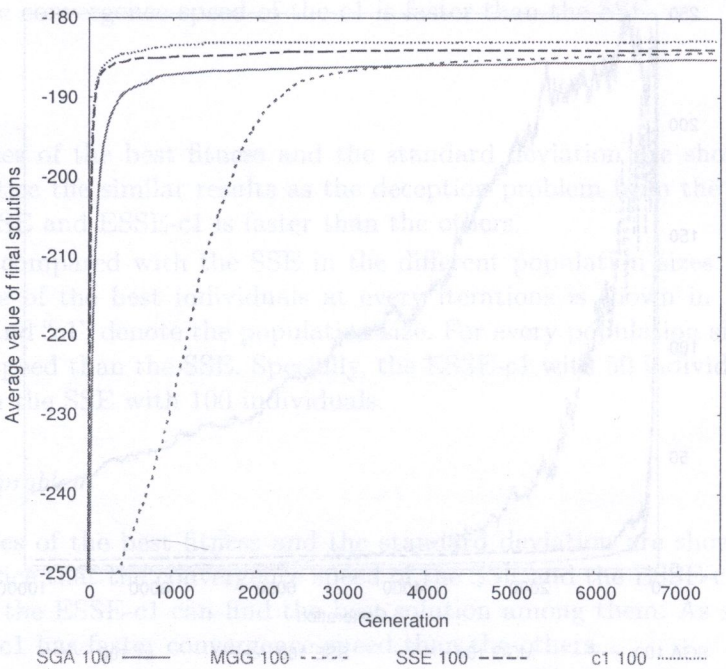

Fig. 9. Average values of solutions of SGA, MGG, SSE and c1 on graph partitioning problem

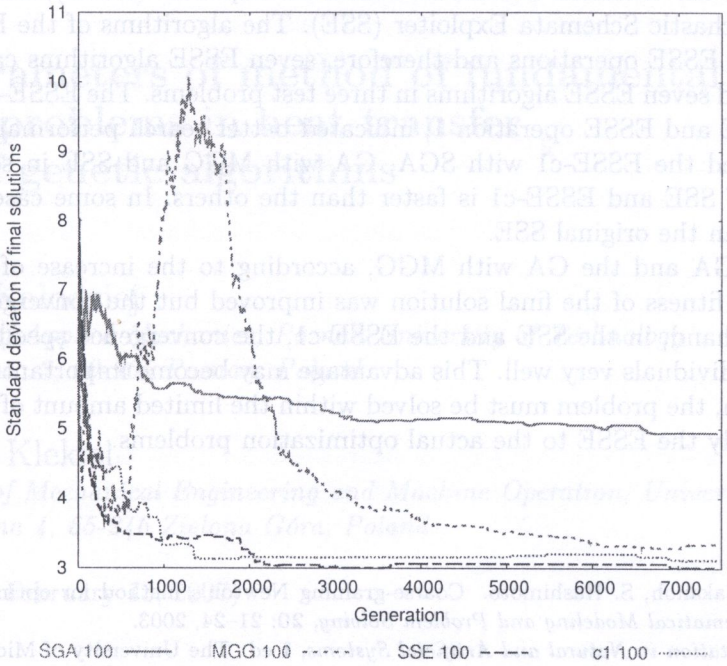

Fig. 10. Standard deviation of solutions of SGA, MGG, SSE and c1 on graph partitioning problem

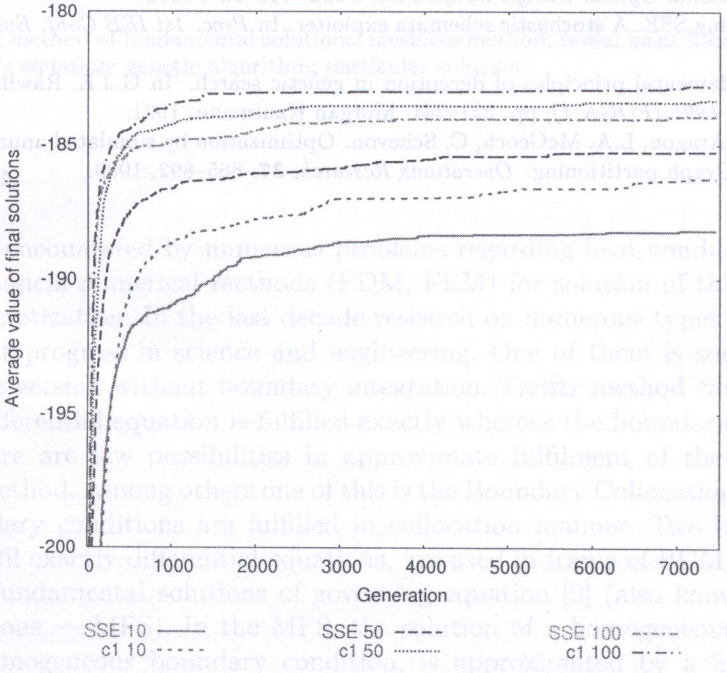

Fig. 11. Average values of solutions of SSE and c1 on graph partitioning problem

5. Conclusions

This paper described the extended Stochastic Schemata Exploiter (ESSE), which is the improved algorithm of the Stochastic Schemata Exploiter (SSE). The algorithms of the ESSE are composed of the SSE and three ESSE operations and therefore, seven ESSE algorithms can be defined.

First, we compared seven ESSE algorithms in three test problems. The ESSE-c1 algorithm, which was composed of SSE and ESSE operation 1, indicated better search performance in all cases.

Next, we compared the ESSE-c1 with SGA, GA with MGG and SSE in some problems. The convergence speed of SSE and ESSE-c1 is faster than the others. In some cases, the speed of the ESSE-c1 is faster than the original SSE.

By the way, in SGA and the GA with MGG, according to the increase of the number of the individuals, the best fitness of the final solution was improved but the convergence speed dropped down. On the other hand, in the SSE and the ESSE-c1, the convergence speed did not depend on the number of the individuals very well. This advantage may become important when, in the actual optimization problem, the problem must be solved within the limited amount of time. In the future, we would like to apply the ESSE to the actual optimization problems.

References

[1] D. Yoshizawa, H. Sakanoh, S. Hashimoto. Coarse-graining Newton's method for optimization (in Japanese). *Trans. IPSJ, Mathematical Modeling and Problem Solving*, **20**: 21–24, 2003.

[2] J.H. Holland. *Adaptation in Natural and Artificial Systems*, 1 ed. The University of Michigan Press, 1975.

[3] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*, 1 ed. Addison Wesley, 1989.

[4] D. Whitly. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials in best. In J.D. Shafer, ed., *Proc. 3rd Int. Conf. Genetic Algorithm*, pp. 116–121. Morgan Kaufmann Pub., 1989.

[5] J.E. Baker. Reducing bias and inefficiency in the selection algorithm. *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 14–21, 1987.

[6] H. Satoh, I. Ono, S. Kobayashi. Minimal generation gap model for gas considering both exploration and exploitation. *Proc. IIZUKA'96*, pp. 494–497, 1996.

[7] I. Ono, S. Kobayashi, K. Yoshida. Global and multi-objective optimization for lens design by real-coded genetic algorithms. *International Optical Design Conference*, **3482**: 110–121, 1998.

[8] N.A. Aizawa. Evolving SSE: A stochastic schemata exploiter. In *Proc. 1st IEE Conf. Evol. Comp.*, pp. 525–529. IEEE, 1994.

[9] L.D. Whitley. Fundamental principles of deception in genetic search. In G.J.E. Rawlins, ed., *Foundations of Genetic Algorithms 1991 (FOGA 1)*, pp. 221–241. Morgan Kaufmann, 1991.

[10] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon. Optimization by simulated annealing: An experimental evaluation. Part I, Graph partitioning. *Operations Research*, **37**: 865–892, 1989.