We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

3

Genetic Algorithm Based on Schemata Theory

Eisuke Kita and Takashi Maruyama
Graduate School of Information Science, Nagoya University
Japan

1. Introduction

In actual complicated optimization problems, it is often difficult to find a global optimum solution in admissible computing time. Therefore, in industrial problems, we should find quasi-optimum solution in admissible computing time. In that case, evolutionary computations (ECs) are very attractive.

There are several algorithms in the EC family; genetic algorithm (GA), evolutionary strategy (ES), genetic programming (GP), and so on(4; 5; 10). Genetic algorithm (GA) has been firstly presented by J.Holland in 1975(5). 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)(4). 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(2; 10). 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 Minimal Generation Gap (MGG)(8). The application of GA with MGG to several actual problems reveals that the GA with MGG is very effective for actual optimization problems(6).

Stochastic Schemata Exploiter (SSE) is also classified into the ECs(1). Although the basic concept of SSE comes from the GA, its algorithm is very different from GA. In GA, the individuals are generated randomly in order to construct a population. After estimating the fitness of individuals, parents are selected from the population according to the fitness value. Offspring are generated from the parents by using genetic operators such as the mutation, the crossover, and so on. SSE algorithm also starts from the population of randomly generated individuals. After estimating the fitness of individuals, sub-populations are generated from the whole population according to semi-order relationship of the sub-populations. Common schemata are extracted from the sub-populations and offspring are generated from the common schemata. The SSE has two attractive features. Firstly, the SSE convergence speed is faster than the SGA because SSE can spread better schemata over the whole population faster than the GA. Secondly, there are very small number of control parameters which has to be defined by users in advance. Since the selection and crossover operators are not necessary in SSE, the control parameters are only population size and mutation rate. However, SSE sometimes converges to not global optimum solution but local one.

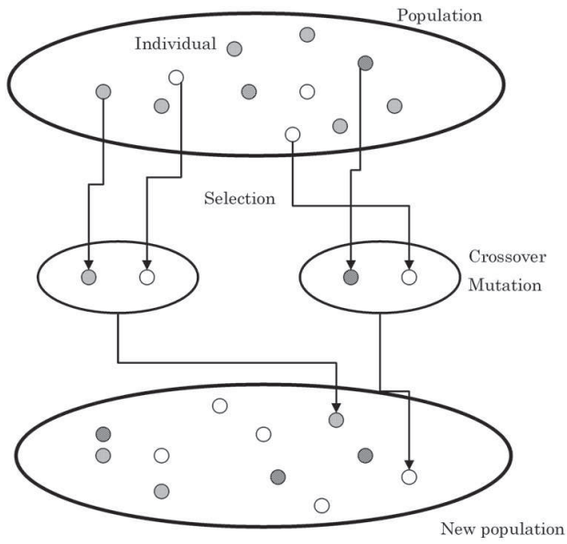


Fig. 1. Simple genetic algorithm (SGA)

The aim of this study is to improve the search performance of SSE without sacrificing the convergence speed. For this purpose, we introduce in this chapter Extended Schemata Exploiter (ESSE) and cross-generational elitist selection SSE (cSSE). In the ESSE, once the common schemata list is defined from the common schemata which are extracted from the individuals in the sub-populations, the list is modified by deleting individual schemata, updating similar schemata and so on. In the cSSE, the cross generational elitist selection(3) is introduced to the original SSE. In the numerical examples, SSE, ESSE and cSSE are compared with genetic algorithm (GA) with minimum generation gap (MGG) and Bayesian Optimization Algorithm (BOA).
The remaining of the chapter is organized as follows. Algorithms of GA, SSE, ESSE, cSSE and BOA are compared briefly in section 2. In sections 3 and 4, the SSE, ESSE and cSSE algorithms are described minutely. Numerical examples are shown in section 5. Some conclusions are summarized again in section 6.

2. Back ground

2.1 Genetic algorithm

Genetic algorithm (GA) was firstly presented by J.Holland in 1975(5). The most fundamental genetic algorithm is often called as simple genetic algorithm (SGA)(4). SGA is illustrated in Fig.1.

1. Construct an initial population by individuals with randomly defined chromosomes.
2. Estimate individuals fitness.
3. Select parents from the population according to the roulette selection of the fitness value.
4. Generate two offspring from the parents by one-point crossover operator.
5. Apply mutation operator to all offspring.
6. Go to step 2 unless convergence criterion is satisfied.

SGA has two disadvantages; the early convergence and evolutionary stagnation(2; 10). The early convergence means that all individuals in the population converge to same local

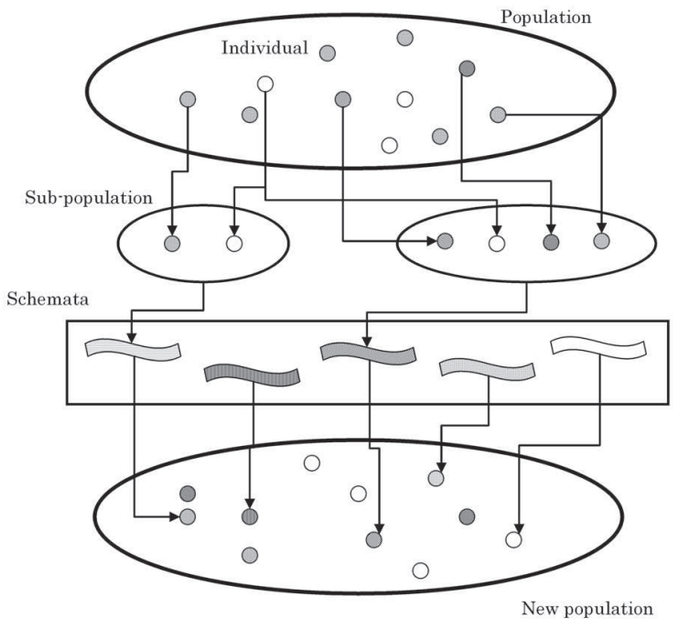


Fig. 2. Stochastic schemata exploiter (SSE)

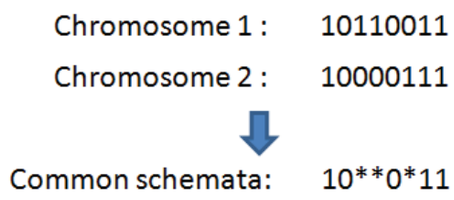


Fig. 3. Example of chromosome and common schema

optimum solutions at early generation and therefore, the global (real) optimum solution cannot be found. The evolutionary stagnation means that the convergence speed slows down at final generations. Minimal Generation Gap (MGG) was presented for overcoming these problems(8). Several numerical results show that the GA with MGG can find better global solutions although the convergence speed is sacrificed(6). The process of the GA with JMMG is summarized as follows.

1. Construct an initial population by individuals with randomly defined chromosomes.
2. Select parents from the population randomly.
3. Generate two offspring from the parents by one-point crossover operator.
4. Estimate fitness of two parents and two offspring.
5. Select the best individual among them.
6. Select the other individual than the best one among them randomly.
7. Replace two selected individuals with parents.
8. Go to step 2 unless convergence criterion is satisfied.

2.2 Stochastic schemata exploiter

The SGA search process can be explained according to the schemata theory. The SGA final goal is to find the set of 0′ and 1′s which represents the optimal solution of the function.

Common set of the individual chromosomes is named as common schemata (Fig3). In the common schema, the uncommon '0's and '1's for all chromosomes are replace with '*'s. As the SGA search process goes, common schemata of the better individuals develops to the binary representation of an optimal solution. If the developing speed of the common schemata can be accelerated, the SGA search performance must be improved well. This is the basic concept of Stochastic Schemata Exploiter (SSE).
The algorithm of SSE is illustrated in Fig.2.

1. Construct an initial population by individuals with randomly defined chromosomes.
2. Estimate individual fitness.
3. Rank individuals according to the descending order of their fitness.
4. Generate sub-populations according to individual rank.
5. Extract common schemata from the individuals in each sub-population.
6. Generate offspring from the extracted schemata.
7. Go to step 2 unless convergence criterion is satisfied.

While SGA generates offspring from parents (individuals), SSE generates from the common schemata extracted from better individuals. Therefore, SSE can accelerate the developing speed of the better common schemata.
The SSE algorithm is described minutely in section 3.

2.3 Extended stochastic schemata exploiter

SSE extracts the common schemata from better individuals. The extracted common schemata are very often identical or very similar schemata. If the identical or very similar common schemata are deleted from the common schemata list, a wider variety of individuals can be generated from the list. This is the basic idea of the extended stochastic schemata exploiter (ESSE).
When two schemata are selected from the list, their similarities are classified as follows:

- **Case 1:** two schemata are identical,
- **Case 2:** one schema is included into the other one,
- **Case 3:** they are partially identical, and
- **Case 4:** they are different.

ESSE operations 1,2 and 3 are defined for the case 1, 2 and 3, respectively.
The ESSE algorithm is composed of the original SSE algorithm and one or more of the above ESSE operations. The ESSE algorithm is illustrated in Fig.4. The different process against SSE is to update the common schemata list by applying the ESSE operations.
The ESSE algorithm is described minutely in section 4.1.

2.4 Cross generational elitist selection SSE

ESSE updates the common schemata list by deleting the identical or similar common schemata from the list in order to improve the search performance. The computational cost for updating the list, however, is relatively expensive.
The aim of cross generational elitist selection SSE (cSSE) is to reduce the computational cost without sacrificing the ESSE search performance. For the purpose, instead of update of the common schemata list in ESSE, the cSSE adopts cross-generational elitist selection(3) and exclusion of identical individuals from a population.
The cSSE algorithm is illustrated in Fig.5. The different process against ESSE is to select better individuals alone from the large population composed of all parents and all offspring in order to define a new population.

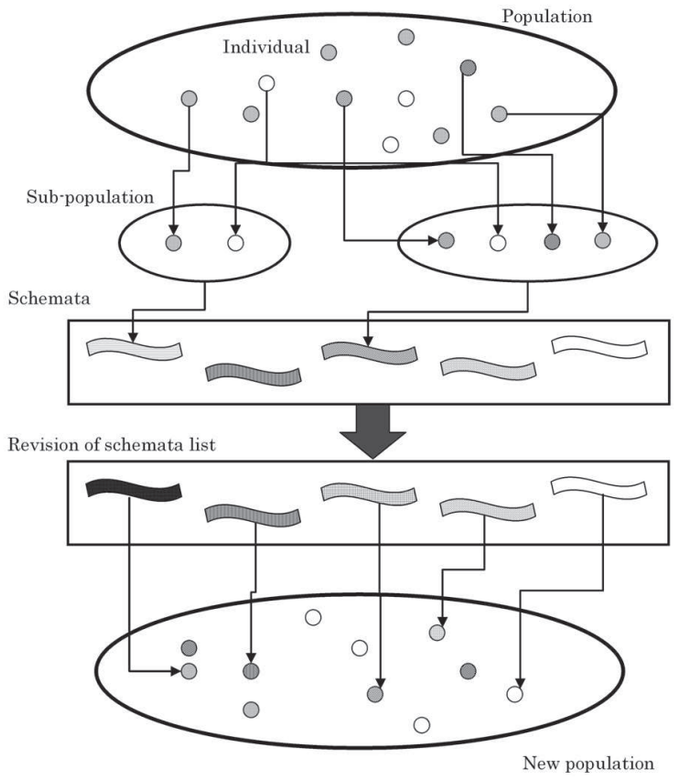


Fig. 4. Extended stochastic schemata exploiter (ESSE)

The cSSE algorithm is described minutely in section 4.2.

2.5 Bayesian optimization algorithm

Estimation of Distribution Algorithm (EDA) is also one of evolutionary computations. The EDA searches a solution according to stochastic model learned from the information of the better solutions in the population. Since offspring are generated from the stochastic model, the selection, the crossover, and the mutation operations are not necessary in EDA.
Bayesian Optimization Algorithm (BOA), which is one of the EDA, was presented by Pelikan et. al.(7). In BOA, the Bayesian network plays as the stochastic model of EDA. The convergence speed of BOA is much faster than that of SGA. In this study, BOA is adopted for confirming the convergence speed of SSE, ESSE and cSSE.

3. Stochastic schemata exploiter

3.1 SSE algorithm

We would like to explain again the process of the stochastic schemata exploiter (SSE).

1. Construct an initial population with randomly generating M individuals.
2. Estimate individual fitness
3. Rank individuals according to the descending order of their fitness.
4. Define M sub-populations according to individual rank.
5. Extract common schemata from individuals in M sub-populations.
6. Generate M offspring from the extracted schemata.

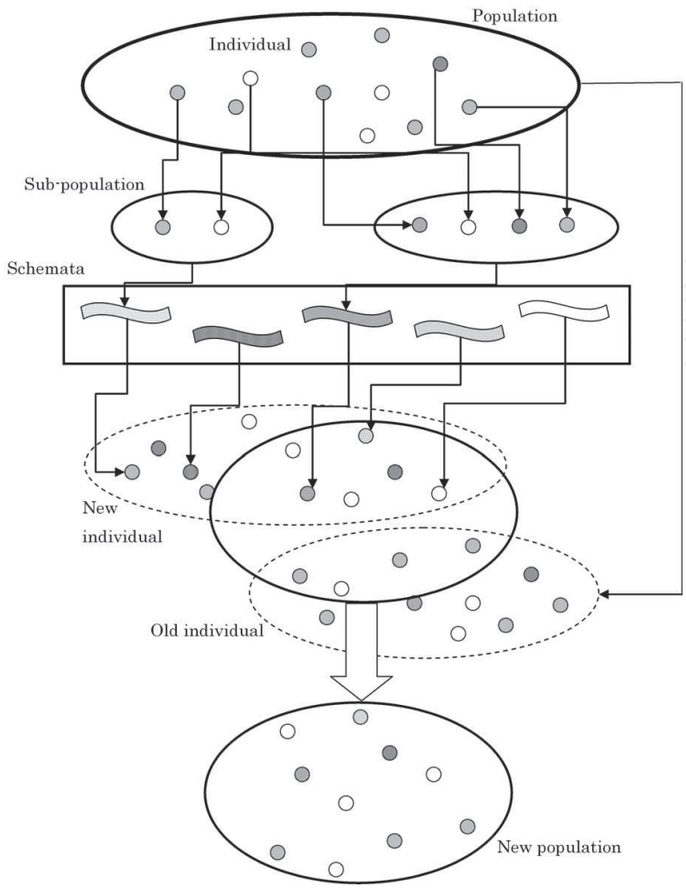


Fig. 5. Cross generational elitist selection stochastic schemata exploiter (cSSE)

7. Go to step 2 unless convergence criterion is satisfied.

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.

3.2 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.

3.2.1 Semi-order relation

The population P is composed of the individuals $c_1, c_2, \cdots$ and c_M, which are numbered according to the descending order of their fitness function. 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

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

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

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

3.2.2 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.

3.3 Extracting common schemata

After defining the sub-populations, the common schemata are extracted as the common set of the chromosome of the individuals in the sub-populations (Fig.3).

3.4 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” (Fig.3).

4. Extended SSE and cross generational elitist selection SSE

4.1 Extended SSE (ESSE)

4.1.1 Algorithm

The ESSE algorithm is illustrated in Fig.4 and summarized as follows.

1. Construct an initial population by individuals with randomly defined chromosomes.
2. Estimate individual fitness.
3. Rank individuals according to the descending order of their fitness.
4. Generate sub-populations according to individual rank.
5. Extract common schemata from the individuals in sub-populations and register them to the common schemata list.
6. Update the common schemata list by applying the ESSE operations.
7. Generate offspring from the extracted schemata.
8. Go to step 2 unless convergence criterion is satisfied.

4.1.2 ESSE operations

The SSE algorithm often generates identical or very similar schemata from different sub-populations. Extended Stochastic Schemata Exploiter (ESSE) updates the common schemata list by applying the ESSE operations.
When the schema A is extracted from the sub-population S_A, the fitness of the schema A is defined as the average fitness of all individuals in the sub-population S_A, which is referred to as $f(S_A)$.
The following operations are applied for two schemata A and B.

4.1.2.1 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 $S_A \cup S_B$.

4.1.2.2 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 $S_A \cup S_B$.

4.1.2.3 ESSE Operation 3

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

1. If $f(S_A) > f(S_B)$, A is kept. If not so, B is kept.
2. A common schema is extracted from $S_A \cup S_B$.

4.1.3 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 $c_1, c_2, \cdots$ and c_7.

c1: Original SSE with ESSE operation 1.
c2: Original SSE with ESSE operations 1 and 2.
c3: Original SSE with ESSE operation 2.

c4: Original SSE with ESSE operations 2 and 3.
c5: Original SSE with ESSE operation 3.
c6: Original SSE with ESSE operations 1 and 3.
c7: Original SSE with ESSE operations 1, 2 and 3.

4.2 Cross generational elitist selection SSE (cSSE)

4.2.1 Cross generational elitist selection

The cross generational elitist selection was presented by Eshelman(3). Algorithm of cross generational elitist selection is summarized as follows.

1. At generation $t-1$, offspring are generated from individuals in the population.
2. Populations of parents and offspring are referred to as $P(t-1)$ and $O(t-1)$, respectively.
3. $P(t-1)$ and $O(t-1)$ are merged to new population $P'(t-1)$. When the sizes of $P(t-1)$ and $O(t-1)$ are M, the size of $P'(t-1)$ is $2M$.
4. Individuals in $P'(t-1)$ are ranked according to their fitness.
5. The population $P(t)$ is generated by selecting M best individuals from $P'(t-1)$.

The original algorithm of cross-generational elitist selection allow multiple identical individuals to exist in a population. However, the algorithm of cross generational elitist selection in cSSE is modified so that identical individuals are deleted from the population.

4.2.2 Algorithm

The cSSE algorithm is illustrated in Fig.5 and summarized as follows.

1. Construct an initial population by individuals with M randomly defined chromosomes.
2. Estimate individual fitness.
3. Rank M individuals according to the descending order of their fitness.
4. Generate M sub-populations according to individual ranks.
5. Extract common schemata from the individuals in M sub-populations.
6. Generate M offspring from M extracted schemata.
7. Delete identical individuals from $2M$ individuals in the population composed of M parents and M offspring.
8. Select M better individuals from the remaining individuals to define new population.
9. Go to step 2 unless convergence criterion is satisfied.

In the above process, the steps 7 and 8 correspond to the cross generational elitist selection.

5. Numerical example

5.1 Test Problems

We would like to compare the algorithm performance in deception and knapsack problems(9).

5.1.1 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

$$f_{deception} = \sum_{i=1}^{n} f_d(x_i) \tag{3}$$

$$x_i \in 0000, 0001, \cdots, 1111$$

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

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

Table 1. Part solutions of deceptive 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 2. Final solutions on deception 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 3. Final solutions on knapsack problem

5.1.2 Knapsack problem

When there are n bag gages 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

$$\begin{aligned} &\max_{\{x_i\}} \sum_{i=1}^{n} c_i x_i \\ &\text{subject to} \sum_{i=1}^{n} a_i x_i \leq b \\ &x_i \in 0,1 \qquad (i = 1, \cdots, n) \end{aligned} \tag{4}$$

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

5.2 ESSE performance evaluation

A two-point crossover is adopted for the SGA and GA with MGG. The crossover rate is 1 (100%). 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. Fifty simulations are performed for each problem from the different initial populations. Their average values are shown in figures.

5.2.1 Comparison of ESSE family

We notice from Tables 2 and 3 that the ESSE-c1, c2 and c6 have the better search performance than the others. The results show that ESSE operation 1 is effective because ESSE-c1, c2 and c6 have the ESSE operation 1. 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 $n_i = 100$. Therefore, the effectiveness of the ESSE operations 2 and 3 may depend on the problem to be solved.

5.2.2 Comparison of ESSE, SSE, SGA, and GA with MGG

The results in the previous section show that the ESSE-c1 is the best among the ESSE family. The ESSE-c1 is compared with SGA, GA with MGG and SSE.

5.2.2.1 Deception problem

Convergence history of the best individual fitness is shown in Fig.6. The figure is plotted with the generation as a horizontal axis and the best individual fitness as a vertical axis, respectively. Numbers following to the algorithm name, in the figures, denote the population size n_i. The population size is determined from numerical experiments as the best value for each algorithm. We notice from figure 6 that the convergence speed of SSE and ESSE-c1 are faster than SGA and GA with MGG and specially, that the ESSE-c1 is the fastest among them. 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.7. In case of the large population size (50 and 100 individuals), the convergence speed of the c1 is faster than the SSE.

5.2.2.2 Knapsack problem

Convergence history of the best individual fitness is shown in Fig.8. Figure is plotted with the generation as a horizontal axis and the best individual fitness as a vertical axis, 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 best individual fitness is shown in Fig.9. In all population sizes, the ESSE-c1 shows faster convergence speed than the SSE. Specially, the ESSE-c1 with 50 individuals can find slightly better solution than the SSE with 100 individuals.

5.3 cSSE Performance evaluation

We would like to compare cSSE with GA with MGG, BOA and SSE.
In GA with MGG, two-point crossover of crossover rate $= 1$ is employed and all individuals are replaced at every generations. In all algorithms, the best mutation rates are determined from numerical experiments.
Maximum generation is 40,000 for deception problem and 15,000 for knapsack problem, respectively. Population size is specified as $N_i = 10, 50, 100$ or 250 for GA with MGG, SSE and cSSE and $n_i = 20, 100, 200$ or 500 for BOA, respectively. Since BOA replaces half population size at every generations, computational cost of fitness function is half as much as the other algorithms. For equalizing the computational cost of all algorithms, the population size of BOA is twice as many as the other algorithms. Simulations are performed 50 times from different initial populations. The average values of the best individual fitness are shown.

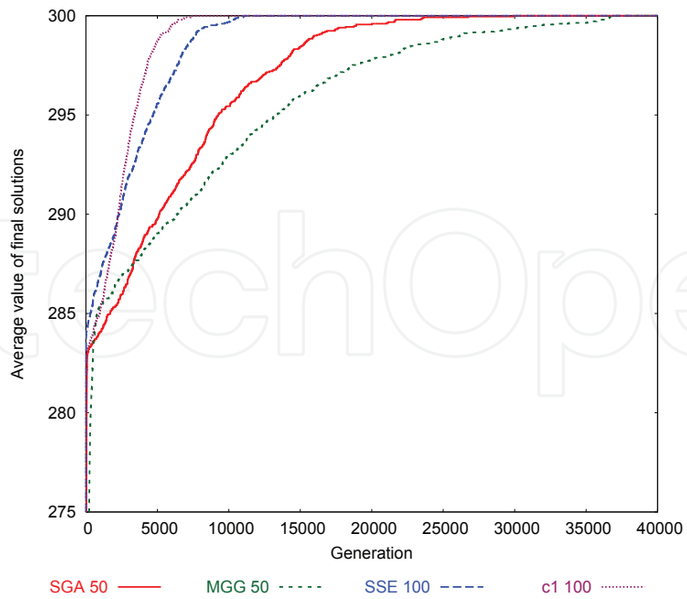


Fig. 6. Comparison of SGA, GA with MGG, SSE and ESSE-c1 in deception problem

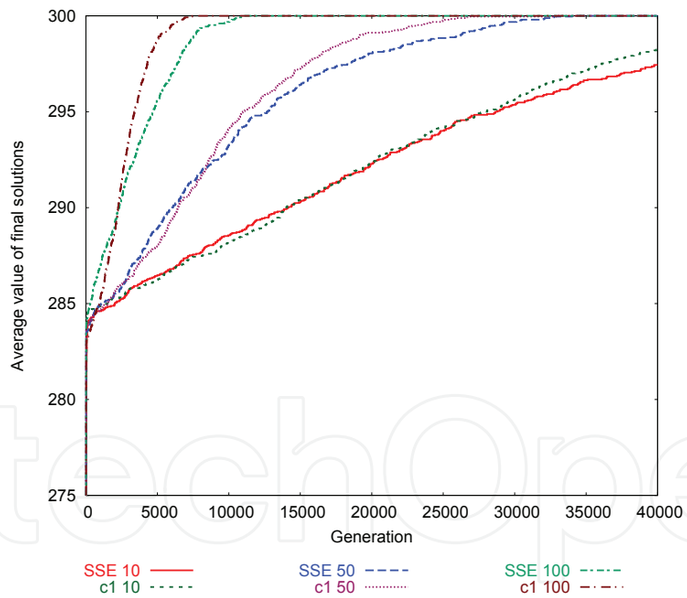


Fig. 7. Comparison of SSE and ESSE-c1 in deception problem

N_i (N_i for BOA)	GA(MGG)	BOA	SSE	cSSE
10 (20)	571.2	533.4	570.0	573.0
50 (100)	574.9	560.4	575.2	588.3
100 (200)	576.1	560.0	586.9	594.6
250 (500)	575.8	560.0	590.7	597.1

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

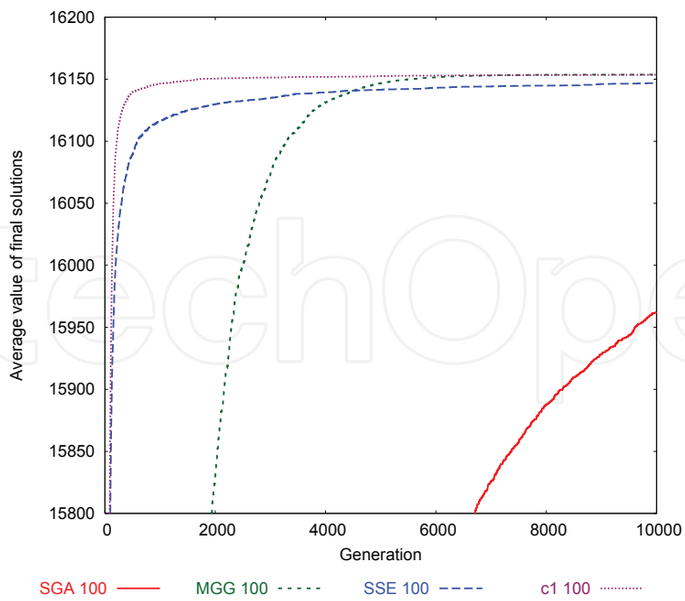


Fig. 8. Comparison of SGA, GA with MGG, SSE and ESSE-c1 in knapsack problem

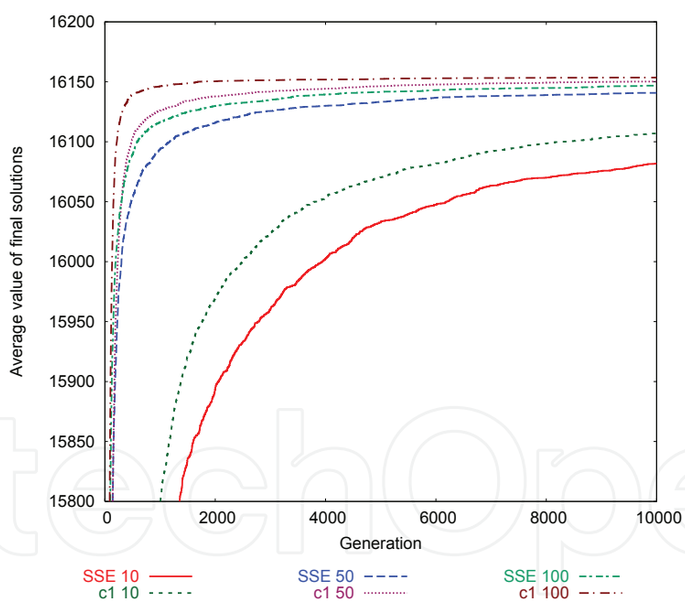


Fig. 9. Comparison of SSE and ESSE-c1 in knapsack problem

5.3.1 Deception problem

Convergence history of the best individual fitness is shown in Fig.10. Figure is plotted with the generation as the horizontal axis and the fitness value as the vertical axis, respectively. Table 2 shows the best individual fitness at final generation.

We notice from Table 4 that the final cSSE solution is the best among them for all cases of population size. Figure 10 illustrates that the cSSE is the fastest among them. Although BOA

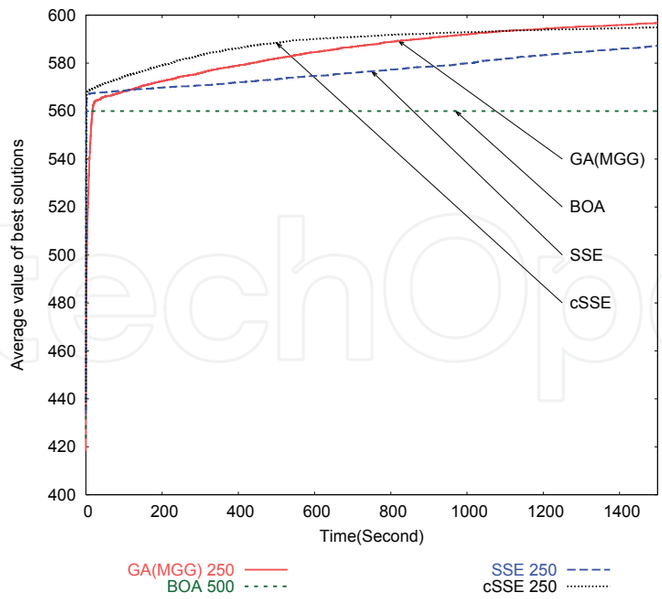


Fig. 10. Comparison of GA with MGG, BOA, SSE and cSSE in deception problem

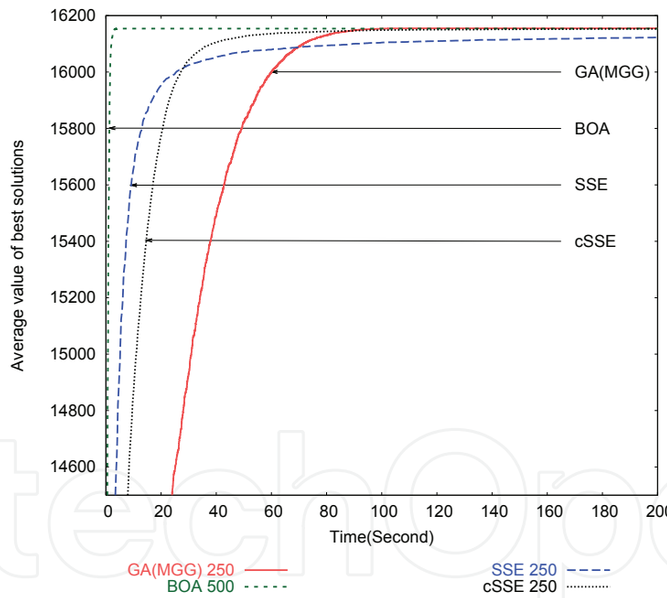


Fig. 11. Comparison of GA with MGG, BOA, SSE and cSSE in knapsack problem

has very fast convergence speed, it may be attracted to a local optimum solution because the final BOA solution is worse than the other.

5.3.2 Knapsack problem

Convergence history of the best individual fitness is shown in Fig.11. We notice that the BOA outperforms the SSE and the cSSE and that the BOA is the best among them from the view-point of both the computational time and the search performance. Remember that the convergence speed of the BOA is generally faster than the GA, SSE and cSSE. Therefore, the

N_i (N_i for BOA)	GA(MGG)	BOA	SSE	cSSE
10 (20)	16060.7	12754.9	16104.4	16141.0
50 (100)	16141.9	15829.3	16142.8	16154.1
100 (200)	16153.8	16111.9	16148.3	16154.8
250 (500)	16155.0	16153.9	16149.5	16155.0

Table 5. Average values of final solutions (Knapsack problem)

above results may denote that the solution space of the Knapsack problem is relatively simple and that there is only one optimal solution in the space. Table 5 shows the fitness value of the best solution at the final generation.
We notice from Table 5 that the final cSSE solution is the best among them. Note that the final cSSE solution at $N_i = 50$ is better than GA solution at $N_i = 100$, BOA solution at $N_i = 500$, and SSE at $N_i = 250$. Figure 11 illustrates that the convergence speed of MGG is the slowest among them.

6. Conclusions

In this chapter, we described stochastic schemata exploiter (SSE) and its improved algorithms such as Extended SSE (ESSE) and cross-generational elitist selection SSE (cSSE).
First, we compared seven ESSE algorithms in test problems. The ESSE-c1 algorithm, which was composed of SSE and ESSE operation 1, showed better search performance than the other ESSE algorithms in the test problems. We compared the ESSE-c1 with SGA, GA with MGG and SSE. 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.
Next, we compared the cSSE with the other algorithms. From the view point of search performance, we notice that cSSE can find slightly or much better solution than the others in all examples. In comparing the convergence speed of algorithms, we notice that the cSSE as well as BOA is fastest among them.

7. References

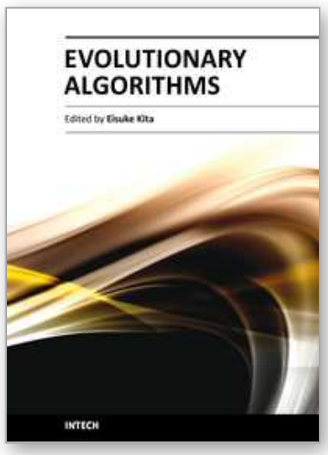
[1] N. A. Aizawa. Evolving SSE: A stochastic schemata explointer. In *Proc. 1st IEE Conf. Evol. Comp.*, pages 525–529. IEEE, 1994.
[2] J.E. Baker. Reducing bias and inefficiency in the selection algorithm. *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, 1987.
[3] L. J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms 1991 (FOGA 1)*, pages 265–283. Morgan Kaufmann., 1991.
[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1 edition, 1989.
[5] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1 edition, 1975.
[6] I. Ono, S. Kobayashi, and K. Yoshida. Global and multi-objective optimization for lens design by real-coded genetic algorithms. *International Optical Design Conference*, 3482:110–121, 1998.
[7] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz. Boa: The bayesian optimization algorithm. In W. Banzhaf, Daida J, A. E. Elben, M. H. Garzon, V. Honavar, M. Jakiela,

and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-1999, San Fransisco, CA)*, pages 525–532. Morgan Kaufmann., 1999.

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

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

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



Evolutionary Algorithms

Edited by Prof. Eisuke Kita

ISBN 978-953-307-171-8
Hard cover, 584 pages
Publisher InTech
Published online 26, April, 2011
Published in print edition April, 2011

Evolutionary algorithms are successively applied to wide optimization problems in the engineering, marketing, operations research, and social science, such as include scheduling, genetics, material selection, structural design and so on. Apart from mathematical optimization problems, evolutionary algorithms have also been used as an experimental framework within biological evolution and natural selection in the field of artificial life.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen