韓國經營科學會誌
第32권 第4號
2007年 12月

적응형 유전알고리즘의 실험적 비교

윤 영 수*

An Experimental Comparison of Adaptive Genetic Algorithms

YoungSu Yun*

■ Abstract ■

In this paper, we develop an adaptive genetic algorithm (aGA). The aGA has an adaptive scheme which can automatically determine the use of local search technique and adaptively regulate the rates of crossover and mutation operations during its search process. For the adaptive scheme, the ratio of degree of dispersion resulting from the various fitness values of the populations at continuous two generations is considered. For the local search technique, an improved iterative hill climbing method is used and incorporated into genetic algorithm (GA) loop.

In order to demonstrate the efficiency of the aGA, i) a canonical GA without any adaptive scheme and ii) several conventional aGAs with various adaptive schemes are also presented. These algorithms, including the aGA, are tested and analyzed each other using various test problems. Numerical results by various measures of performance show that the proposed aGA outperforms the conventional algorithms.

Keyword : Adaptive Genetic Algorithm, Adaptive Scheme, Local Search Technique

1. Introduction

During past two decades, a lot of methodologies to enhance the performance of genetic algorithm (GA) have been developed. Especially, various works that regulate genetic parameters such as crossover rate, mutation rate, population size, etc., have been performed by many re-

논문접수일 : 2007년 02월 22일 논문게재확정일 : 2007년 10월 05일

* 조선대학교 경상대학 경영학부

searchers [5, 6, 9, 10, 12, 20, 27,]. Most of the works can adaptively regulate the rates of various genetic parameters, and the logics used for adaptive regulation usually consider a balance between exploration and exploitation during genetic search process.

Exploration investigates new and unknown areas in search space, while exploitation uses the knowledge acquired by exploration to reach better positions on search space. Therefore, how to set genetic parameters can be the most important factor in the determination of exploration versus exploitation tradeoff. It has long been acknowledged that these two properties have greatly influence on the performance of GA [9]. If the poor setting of genetic parameters is used, a correct balance between exploration and exploitation in GA search process can not be reached in a desired way, which may lead GA search to premature convergence at a local optimum. Therefore, the performance of GA is highly affected by how to balance its exploration and exploitation.

Designing a correct setting of genetic parameters is not an easy task, since the interaction among them has not been known completely. Furthermore, the values of different genetic parameters may be necessary during genetic search process to keep a correct balance between exploration and exploitation [10]. To overcome this weakness of GA, various adaptive GAs (aGAs) have been developed [2, 6, 7, 10, 22, 27]. Most of them adaptively regulate genetic parameters during the course of genetic search, and their objectives are to set a correct balance between exploration and exploitation.

Especially, some idea for adaptively regulating the rates of crossover and mutation operations in GA has been suggested in many studies [10, 12, 13, 16, 20, 24, 25]. These studies use various heuristics or mathematical formulations, and the genetic parameters controlled in each algorithm are adaptively regulated during genetic search process. Therefore, much time for fine-tuning the parameters can be saved, and the search ability of GA can be also improved in finding global optimal solution. However, most of the adaptive schemes used in the conventional studies depend highly on the problem under consideration, that is, different adaptive parameters are required for different problems.

In this paper, therefore, we develop an efficient algorithm, called the adaptive genetic algorithm (aGA). The developed aGA has the adaptive scheme which can automatically determine the use of local search technique and adaptively regulates the rates of crossover and mutation operations during genetic search process. In Section 2, some concepts and logics for keeping a correct balance between exploitation and exploration in GA search are suggested. By the concepts and logics, the aGA is developed in Section 3. To prove the efficiency of the aGA, canonical GA without any adaptive scheme and several conventional aGAs with various adaptive schemes taken from some previous works are also presented, and all the algorithms are then tested and analyzed using various test problems in Section 4. Some conclusion and remarks are followed in Section 5.

2. Exploitation and Exploration in GA search

Keeping a correct balance between exploitation and exploration in GA search has been known as a very difficult task [6, 10, 20].

In GA search process, a regulation between exploitation and exploration is usually determined according to the behaviors of the individuals of GA population. As GA is converging, the similarity among the individuals of GA population is increased and the variance of the fitness values of the individuals is thus decreased. If the GA search proceeds to the global optimal solution, then it can get a reliable solution. However, it may also reach to premature convergence, if it gets stuck at a local optimal solution, which makes the improvement of solution and the search to global optimal solution very difficult.

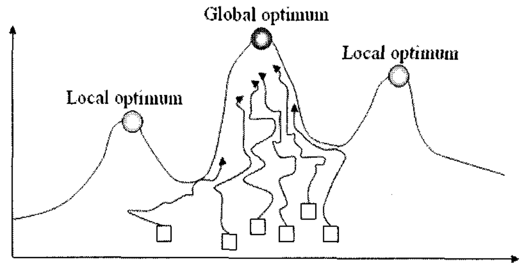
A reliable way in GA search requires the abilities that i) continuously keep good individuals with high fitness values in order to guide the convergence toward global optimal solution as well as ii) suitably maintain the diversity of individuals in order to avoid premature convergence toward local optimal solutions, which is usually called as exploitation and exploration trade-off. Therefore, both exploitation and exploration should be suitably controlled during genetic search process to preserve good individuals and maintain their diversity, which eventually enhances GA search ability to global optimal solution.

By exploration, the diversity of individuals in GA population is maintained, and its premature convergence to local optimal solutions can be also avoided. On the other hand, by exploitation, a possibility of finding respective individuals for GA population is increased, and the searching ability to global optimal solution is enhanced.

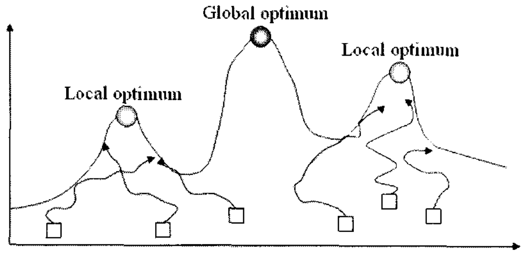
Many studies have been performed for keeping a correct balance between exploitation and exploration in GA search by adaptively regulating the rates of crossover and mutation operations. Srinivas and Patnaik [20] developed a scheme that the rates of crossover and mutation operations are increased when GA population tends to get stuck at a local optimal solution and are also decreased when the population is scattered in the search space of GA, in order to maintain the diversity of individuals and reinforce the search to global optimal solution. Wu et al. [25] suggested an improved scheme for reinforcing the ability of the mutation operation used in Srinivas and Patnaik [20]. The improved scheme inserts an additional logic into the original mutation scheme in order to prevent the deterioration of the performance of GA.

Mak et al. [16] proposed a scheme to regulate adaptively the rates of crossover and mutation operations according to the performance of genetic operators. It increases the occurrence rates of crossover and mutation operations, if it consistently produces better individuals from previous generation to current generation; however, it also reduces the occurrence rates, if it always produces poorer individuals. Herrena and Lozano [10] suggested that i) the rate of crossover operation should be increased for enhancing the search ability to global optimal solution, ii) the more individuals with high fitness values should be chosen for crossover operation to avoid premature convergence to local optimal solutions, and iii) the rate of crossover operation must be decreased for keeping respective individuals in GA population. Yun and Moon [23] controlled the rates of crossover and mutation operations to be increased, if they consistently produce better offspring during continuous two generations of GA; however, the rates are also decreased if they consistently produce poorer offspring during the generations. This scheme is based on the fact that it encourages the well-performing operators to produce much better offspring, while also reduces the generating chance for poorly performing operators to destroy the potential individuals, during genetic search process.

The common characteristics of these studies mentioned above are classified into two situations: GA is converging and not converging. These two situations can be summarized in [Figure 1] and [Figure 2]. If GA is converging like [Figure 1], then the rates of crossover and mutation operations should be increased for both preventing premature convergence and enhancing the search to global optimal solution. On the other hand, if GA is not conversing like [Figure 2], then the rates of crossover and mutation operations should be decreased to avoid the introduction of new individuals with poor fitness values and guide the search toward convergence.



[Figure 1] The situation that GA is converging



[Figure 2] The situation that GA is not converging

However, there are some weaknesses in GA search strategies for the above two situations.

For the situation of [Figure 1], as GA is converging, the similarity among the individuals of GA population becomes higher. Therefore, the fitness values of the individuals are significantly similar to each other, and the variety of the population is reduced, which definitely deteriorate the performance of GA. Considering this situation, it may be difficult for GA search to avoid premature convergence even though the rates of crossover and mutation operations are increased, since crossover and mutation operations using the individuals with similar fitness values may make the introduction of new individuals difficult and also prevent GA from converging to global optimal solution.

The technique that helps improving this weakness of GA performance is to forbid the introduction of an individual into the population when too many similar individuals already exist in it. One possible alternative is to insert new individuals into current population. These new individuals should not have the same similarity as the current population and also keep a certain high fitness values that are similar to that of the population. The mutation operator in GA is an example of such alternatives, but it usually generates undesired random outputs. Local search technique which can explore around the convergence area after GA search at each generation is a possible alternative, since it is capable of producing new individuals with a certain high fitness value like the good individuals generated by GA [5].

By this way, if we consider both the use of a local search technique and the case of increasing the rates of crossover and mutation operations when GA is converging, then the GA performance by this combined strategy may be superior to that by the latter alone. By local search technique, the introduction of new individuals with certain high fitness values will be continuous. By the increase of the rates of crossover and mutation operations, GA search to global optimal solution can be reinforced.

In the situation of [Figure 2] that GA is not conversing, it can be divided into two ways : one is to consider the situation that the current fitness value of GA is inferior to the previous fitness ones during genetic search process, another is that the fitness values of GA in continuous generations do not show any change at all.

By the first situation, the introduction of the inferior individuals in current generation will be increased to the next generation, which definitely deteriorates the performance of GA. To overcome this weakness, the introduction of the new individuals should be reduced in constructing new population of the next generation. Decreasing the rates of crossover and mutation operations is a good alternative for this purpose. By decreasing the rates, the good individuals with high fitness values in current generation are kept, which can help the convergence of solution.

The second situation often occurs after GA search has been significantly progressed. At that time, any improvement or convergence of the solution may not be occurred, even though the introduction of new individuals to next generation is reduced by decreasing the rates of crossover and mutation operations, which is mainly because the fitness values of the individuals at that time are significantly similar to each other. Therefore, a new technique for improving this situation is required. A possible alternative is to insert the new individuals, resulting from the local search technique which is already explained when GA is converging, into current population.

With the improved methods stated above, we can reach the improved three situations as shown in <Table 1>. To realize the three situations in <Table 1>, we use the degree of dispersion $C(t)$of the fitness values of population at generation t as follows :

$$C(t)=\frac{f_{\max}(t)-\bar{f}(t)}{f_{\max}(t)-f_{\min}(t)} \tag{1}$$

where $f_{\max}(t)$ and $f_{\min}(t)$ are the maximal and minimal fitness values of the population, respectively, and $\bar{f}(t)$ is the average of all the fitness values of the population, at generation t.

The $C(t)$in equation (1) can confirm the convergence situation of solution in GA population [10]. Using the equation (1), the ratio of degree

<Table 1> Improved three situations

Situation 1	If GA is converging, we increase the rates of crossover and mutation operations and also apply a local search technique to GA loop. By this way, both the reservation of good individuals and the introduction of new individuals can be simultaneously achieved.
Situation 2	If GA is not converging (the current fitness value of GA is inferior to the previous fitness value), we decrease the rates of crossover and mutation operations. By this way, the introduction of new individuals is reduced and the convergence of solution can be progressed.
Situation 3	If GA is not converging (the fitness values in continuous generations of GA do not show any change at all), we apply a local search technique to GA loop. By this way, several new individuals, both not having the similarity of the current population and also keeping a certain high fitness value that is similar to that of the population, are generated and inserted into GA loop.

of dispersion (CR) of the fitness values in continuous two generations can be formulated as follow :

$$CR = \frac{C(t)}{C(t-1)} \tag{2}$$

Using the CR in equation (2), the improved three situations in <Table 1>, when minimization is assumed, can be represented as shown in <Table 2>.

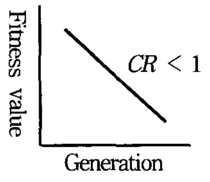
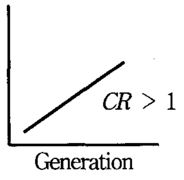
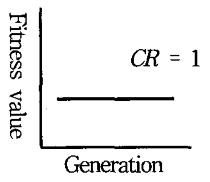
〈Table 2〉 Three conditions

	Conditions
Situation 1	$CR < 1$
Situation 2	$CR > 1$
Situation 3	$CR = 1$

Using <Table 1> and <Table 2>, we can formulate the procedure to adaptively regulate a balance between exploration and exploitation during GA search. The procedure by using the rate of crossover operation (p_C), that of mutation operation (p_M) and local search technique is shown in [Figure 3].

```
Procedure : Regulation of a balance between
            exploitation and exploration
  begin
      if $CR<1$ then
          $p_C(t+1) = p_C(t) + 0.1$;
          $p_M(t+1) = p_M(t) + 0.01$;
          apply local search technique to GA
          loop;
      if $CR>1$ then
          $p_C(t+1) = p_C(t) - 0.1$;
          $p_M(t+1) = p_M(t) - 0.01$;
      if $CR=1$ then
          apply local search technique to
          GA loop;
      end
end
```

[Figure 3] Regulation of a Balance Between Exploitation and Exploration

Condition	p_C	p_M	Local Search technique
	↑	↑	○
	↓	↓	-
	-	-	○

[Figure 4] Regulation of p_C, p_M and local search technique in GA search

[Figure 4] summarizes the procedure defined in [Figure 3]. The detailed approaches using GA and local search technique for implementing the proposed procedure are presented in Section 3.

In [Figure 4] the arrow sighs ↑ and ↓ of the p_C and p_M mean that the rates of the crossover and mutation operations are respectively increasing and decreasing during genetic search process. The circle sigh (O) of the local search is to apply local search technique to GA loop.

3. Design of aGA

For designing the aGA, both GA and local search technique are used. The former is to globally search the whole search space, while, the latter is to locally search around the convergence area in GA loop. By the mixed use of two approaches, a balance between exploitation and exploration during the aGA search process can be adaptively regulated. The detailed procedures for implementing both the GA and local search approaches are shown in the following subsections.

3.1 GA Approach

For the representation of GA, we use a real-number representation instead of a bit-string one, since the former has several advantages of (i) being better adapted to many optimization problems, (ii) speeding up the search against bit-string representation, and (iii) making the development of the approaches easy for hybridizing with conventional local search techniques [3]. The GA is used as a main algorithm of the proposed aGA. For the GA, the initial population is made by random search. Three genetic operators of selection, crossover, and mutation use the elitist strategy in enlarged sampling space [8], uniform arithmetic crossover operator [17] and uniform mutation operator [17], respectively. The offspring resulting from crossover and mutation should satisfy the constraints which are used in each problem under consideration. If some individuals of the offspring do not satisfy the constraints, they are rejected in the selection stage for the next generation. The detailed implementation procedure is as follows:

Step 1 : Initial population.
We use the population resulting from random search within all feasible search spaces.

Step 2 : Genetic operators.
Selection : elitist strategy in enlarged sampling space [8].
Crossover : uniform arithmetic crossover operator [17].
Mutation : uniform mutation operator [17].

Step 3 : Fitness test.
Do fitness test using the offspring satisfying constraints.

Step 4 : Stop condition.
If a pre-defined maximum generation number is reached or a global optimal solution is located during genetic search process, then stop; otherwise, go to Step 2.

3.2 Local Search Approach

Local search techniques usually use local information about the current set of data (state) to determine a promising direction for moving some of the data set, which is in turn used to form the next set of data. The advantage of local search techniques is that they are simple and computationally efficient. However, they are easily entrapped in a local optimum. In contrast, global search techniques such as GA explore the global search space without using local information about promising search direction. Consequently, they are less likely to be trapped in local optima, but their computational cost is higher.

Many researchers have reported that the hybrid approaches both with GA and local search technique produces certain benefits [14, 15, 21]. The reason is that the hybrid approaches can combine a merit of GA with that of local search technique. That is, hybrid approaches are less likely to be trapped in a local optimum than local search technique. Due to local search technique, hybrid approach often converges faster than the GA does.

In our study, therefore, we use a method to hybridize GA with local search technique. This approach seen in most of conventional hybrid GAs is to incorporate local search technique into GA loop [8, 21]. With this approach, local search technique is applied to each newly generated offspring to move it to a local optimum before injecting it into the new population.

For the proposed aGA, we use the iterative hill climbing method suggested by Michalewicz [17] and improve it. This method can guarantee the desired properties of local search technique for hybridization as explained above. The main difference between the conventional iterative hill climbing method and the improved iterative hill climbing method is that the latter selects an optimal individual among the individuals satisfying the constraints of the hybrid GA, while, the former selects a current individual at random, which allows the latter to have various search abilities and good solutions unmet by the former. The detailed procedure of the improved iterative hill climbing method, when minimization is assumed, is shown in [Figure 5].

```
Procedure: improved iterative hill climbing
method in GA loop
  begin
    Select an optimum individual $v_c$ in current
      GA loop;
    Generate randomly as many individuals as
      the population size in the neighborhood of
      $v_c$ ;
    Select an individual $v_n$ with the optimal
      value of the objective function $f$ among
      the individuals newly generated;
    if $f(v_c) > f(v_n)$ then
      $v_c \leftarrow v_n$
    end
  End
```

[Figure 5] Procedure of the improved iterative hill climbing method

3.3 Implementing the GA

With the GA and local search approaches proposed in Sections 3.1 and 3.2, we design the aGA. The detailed procedure for its implementation is as follows:

Steps 1-3 : apply the same steps 1, 2 and 3 as the GA approach of Section 3.1.

Step 4 : Stop condition.
If a pre-defined maximum generation number is reached or a global optimal solution is located during genetic search process, then stop all steps.

Step 5 : Regulation of a balance between exploitation and exploration.
Calculate *CR* using equations (1) and (2) and then adapt the procedure of [Figure 3] Go to step 2.

4. Numerical Examples

In this Section, two test suits both with simple search space and complex search space are used to compare the performance of the developed aGA. The result obtained by the aGA are analyzed and compared with other conventional algorithms. The algorithms for experimental comparison are taken from several conventional works [16, 20, 22, 25]. Each algorithm is summarized in <Table 3>.

In <Table 3>, the GA uses the same GA procedure with Section 3.1. The aGA-1 employed the fitness values of parent and offspring at each generation in order to construct an adaptive scheme by regulating the rates of crossover and mutation operations. The aGA-2 developed an adaptive scheme using various fitness values at each generation of GA, and the improved scheme of the p_M used in the aGA-2 was applied to the aGA-3. The last algorithm for comparison, the aGA-FLC used a fuzzy logic controller (FLC), and its main scheme is to use two FLCs (crossover FLC and mutation FLC), which are implemented independently to adaptively regulate the rates of crossover and mutation operations using the changes of average fitness in GA population.

<Table 3> Conventional algorithms for experiment comparison

Algorithm	Feature
GA	Canonical GA
aGA-1	Adaptive regulation by heuristic procedure[16]
aGA-2	Adaptive regulation by heuristic condition[20]
aGA-3	Adaptive regulation by the improved scheme of the p_M used in aGA-2[25]
aGA-FLC	Adaptive regulation by fuzzy logic controller[22]

The above-stated algorithms (aGA-1, aGA-2, aGA-3 and aGA-FLC) and the proposed aGA are all controlling the rates of crossover and mutation operations and also keeping a balance between exploitation and exploration during their search processes.

For experimental comparison under a same condition, the parameters of each algorithm are set at population size : 20, crossover rate : 0.5, mutation rate : 0.05, maximum generation number : 2,000. The neighborhood search range for the improved iterative hill climbing method used in local search is 1.0.

The "crossover rate" and "mutation rate" for the GA loop are fixed at same values during its search process. However, the rates for the conventional four algorithms (aGA-1, aGA-2, aGA-3 and aGA-FLC) and our proposed aGA are adaptively regulated during their search processes, respectively. Altogether 20 iterations are executed to eliminate the randomness of the searches. The procedures of all the algorithms are implemented in Visual Basic language under IBM-PC Pentium-863Mhz computer with 512 MByte RAM.

For comparing the performance between the aGA and the other algorithms, various measures are used in each test suit. The measures used are shown in <Table 4>.

<Table 4> Measures of performance

Notation	Description
BFV	Best fitness value
AFV	Average fitness value
NGS	Total number of getting stuck at a local optimum
CPU	Average CPU times(Unit : Sec.)
TGN	Total generation number

In <Table 4>, the BFV and AFV are obtained after an algorithm reaches to a given stop condition, respectively. The NGS means the total number that an algorithm gets stuck at a local optimum. The CPU and TGN imply the average CPU time and the total number of generations, when an algorithm reaches to a given stop condition, respectively. Among the measures, the NGS and TGN are only obtained when the global optimal solution was already known, since, in the test problem which the global optimal solution was unknown, the NGS is not able to be obtained and the TGN has no meaning.

4.1 Test suit 1

Experiments on the test suit 1, summarized in <Table 5>, have been carried out in order to compare the performances of the conventional algorithms and the proposed aGA.

<Table 5> Test suit 1 with global optimal solution

Test function	Global optimal Solution
Rastrign function(f_{Ras})	0*
Binary function(f_{Bin})	0
Rosenbrock function(f_{Ros})	0
Reliability function(f_{Rol})	0

* Global optimal solution was already known

The detailed descriptions on each algorithm of test suit 1 are as follows:

- *Rastrign functions* (f_{Ras}) [11]: this considers a function with five continuous variables, its global minimum was $f_{Ras}(x^*) = 0$ at $x_1 = x_2$ $x_3 = x_4 = x_5 = 0$, and all of the variables should be considered as continuous values within the range -5.12 to +5.12. The mathmetical formulation is as follows:

$$f_{Ras}(x_1, x_2, x_3, x_4, x_5) = 15 + \sum_{i=1}^{5}(x_i^2 - 3\cos(2\pi x_i))$$

- *Binary function* (f_{Bin}) [3]: this is a multi-model function of two continuous variables and reaches its global maximum was $f_{Bin}(x^*)$ = 1.0 at $x_1 = x_2 = 0$. The search ranges of x_1 and x_2 have the range -100.0 to +100.0, respectively. The expression is as follows:

$$f_{Bin}(x_1, x_2) = 0.5 - \frac{(\sin\sqrt{x_1^2 + x_2^2})^2 - 0.5}{1.0 + 0.001(x_1^2 + x_2^2)^2}$$

- *Rosenbrock function* (f_{Ros}) [4]: this is to minimize a function of two variables and has a global minimum of $f_{Ros}(x^*) = 0$ at $x_1 = x_2 = 1.0$, and each of x_1 and x_2 have the continuous values within the range -2.048 to +2.048. The expression is as follows:

$$f_{Ros}(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$$

- *Reliability Optimization Problem* (f_{Rol}) [18] : this problem is an optimal redundancy allocation problem with the 15 complex component systems. The mathematical formulation is as follows:

$$\begin{aligned} &\textit{maximize } f_{Rel}(x) = \prod_{j=1}^{15}\{1 - (1 - r_j)^{xj}\} \\ &\textit{subject to } g_{1(x)} = \sum_{j=1}^{15}(c_j \cdot x_j) \le 400 \\ &\qquad g_2(x) = \sum_{j=1}^{15}(w_j \cdot x_j) \le 414 \\ &\qquad x_j^L \le x_j \le x_j^U : \textit{integer } j = 1, \cdots, 15 \end{aligned}$$

where r_j, c_j and w_j are coefficient and their values are appeared in [18].

The global optimum was known as x = [3 4 6 5 3 2 4 5 4 2 3 4 5 4 5] with the objective value f_{Rel} = 0.9456, $g_1(x)$ = 392, and $g_2(x)$ = 414.

Using the four test functions described above, we tested the proposed aGA and the conventional algorithms (GA, aGA-1, aGA-2, aGA-3 and aGA-FLC). The results obtained are summarized in <Table 6> For each test function, the analysis is as follows.

i) f_{Ras}

In terms of the BFV, the algorithms (GA, 〈Table 6〉 Computational results for test suit 1

	f_{Ras}					f_{Bin}				
	BFV	AFV	NGS	CPU	TGN	BFV	AFV	NGS	CPU	TGN
GA	0.0000	0.0025	18	5.0338	1,886	1.0000	0.9566	15	2.2546	1,735
aGA-1	0.7417	4.3884	20	4.4079	2,000	1.0000	0.9566	15	2.3735	1,735
aGA-2	0.0000	0.0009	16	3.9537	1,812	1.0000	0.9628	18	2.9673	1,906
aGA-3	0.0000	0.0014	13	6.4178	1,786	1.0000	0.9737	14	2.9818	1,628
aGA-FLC	0.0276	0.1484	20	4.5291	2,000	0.9978	0.9628	20	1.9518	2,000
aGA	0.0000	0.0006	13	4.5020	1,625	1.0000	0.9782	10	2.3033	1,359

	f_{Ros}					f_{Rel}				
	BFV	AFV	NGS	CPU	TGN	BFV	AFV	NGS	CPU	TGN
GA	0.0000	0.0059	19	2.6133	1,901	0.9228	0.8844	20	09.7535	2,000
aGA-1	0.0001	0.1755	20	2.5397	2,000	0.9240	0.8901	20	09.1447	2,000
aGA-2	0.0000	0.0106	17	2.5442	1,738	0.9244	0.8920	20	10.2288	2,000
aGA-3	0.0000	0.0087	18	2.9112	1,827	0.9233	0.8900	20	09.3555	2,000
aGA-FLC	0.0001	0.1281	20	2.3755	2,000	0.9339	0.8983	20	05.6521	2,000
aGA	0.0000	0.0001	12	2.3940	1,571	0.9465	0.9445	16	10.6914	1,966

aGA-2, aGA-3 and aGA), except for the aGA-1 and aGA-FLC, located the global optimal solution, which means that the adaptive schemes used in the aGA-1 and aGA-FLC do not well control the rates of crossover and mutation operations. The result of the BFV also affects that of the AFV, that is, the aGA-1 shows the worst performance.

The convergence ability to global optimal solution in each algorithm can be confirmed by the comparison of the total number of getting stuck at a local optimum. In terms of the NGS, the abilities of the aGA-3 and aGA show the best performance. However, the aGA-1 and aGA-FLC do not show any convergence to the global optimal solution at all, which means that the adaptive schemes used in the aGA-3 and aGA outperform those in the aGA-1 and aGA-FLC.

In terms of the CPU, since all the adaptive algorithms have additional search schemes, they, except for the aGA-3, have slightly faster running time than the GA. This means that their adaptive schemes can guide their searches to the global optimal solution with the lower rates of crossover and mutation operations than those in the GA.

Similar results such as the BFV and NGS are also shown in terms of the TGN. The aGA-1 and aGA-FLC are the slowest performers because they always get stuck at a local optimum in all trials, which is also proved in terms of the NGS.

ii) f_{Bin}

All the algorithms, except for the aGA-FLC, located the global optimal solution in terms of the BFV. However, in the comparison of the AFV, the aGA is the best performer even though all the algorithms have same value in the BFV.

In the comparison of the NGS and TGN, the performances of the aGA are significantly superior to the remainders, which imply that the adaptive scheme used in the aGA is well controlling the rates of the crossover and mutation operations rather than the other adaptive algorithms (GA-1, aGA-2, aGA-3 and aGA-FLC), during the course of search. However, the aGA-FLC, especially, shows the worst performance in terms of the BFV, NGS and TGN. This can be analyzed that the FLC used for regulating the rates of crossover and mutation operations does not well control the rates.

iii) f_{Ros}

In terms of the BFV, AFV, NGS and TGN, the GA shows the better performances than aGA-1 and aGA-FLC, even though the latter have additional schemes to control the rates of crossover and mutation operations. However, the aGA shows the best performance in all the measures of performance, except for the CPU.

The results of the NGS and TGN mean that the aGA-1 and aGA-FLC do not control the rates of crossover and mutation operations at all, which guides the algorithms toward local optimal solutions instead of the global optimal solution. These results are very similar to those of the f_{Ras}.

iv) f_{Rel}

This test function has more complicated inequality constraints than the f_{Ras}, f_{Bin} and f_{Ros}. It means that locating the global optimal solution in each algorithm become much more difficult than those in the f_{Ras}, f_{Bin} and f_{Ros}. Therefore, all the algorithms, except for the aGA, failed to locate the global optimal solution. This is proved in terms of the NGS and TGN.

Based on all the comparisons of the <Table 6>, each algorithm shows various performances according to the problem under consideration. It means that an algorithm shows not only the best performance in a test function, but also the worst performance in other test functions. However, the aGA developed in this paper shows considerately better performances in most of the test functions shown in <Table 6> than the other competing algorithms. This means that the adaptive scheme and the local search technique used for regulating a balance between the exploitation and exploration in the aGA are well guiding the search toward the global optimal solution.

4.2 Test suit 2

For more various comparisons with the aGA, the conventional GA and several aGAs, four types of engineering optimization problems without the global optimal solutions are used here. These problems are taken from several conventional works. <Table 7> summarizes their descriptions.

<Table 7> Test suit 2 without global optimal solution

Test problem	Global optimal Solution
Coil Compression Spring Problem(f_{Coi})	X*
Pressure Vessel Problem(f_{Pre})	X
Reinforced Concrete Beam Problem(f_{Rei})	X
Gear Train Problem(f_{Gea})	X

* Global optimal solution has not been known

The detailed descriptions on each test problem of test suit 2 are as follows:

- *Coil compression problem* (f_{Coi}): this problem was applied by Wu and Chow [26], and its mathematical formulation is as follows:

$$\textit{minimize} \quad f_{Col}(x) = \pi^2 x_2 x_3^2 (x_1 + 2)/4$$

subject to

$g_1(x) = (8C_f F_{\max}\, x_2/\pi x_3^3) - s \le 0$

$g_2(x) = l_f - l_{\max} \le 0$

$g_3(x) = d_{\min} - x_3 \le 0$

$g_4(x) = x_2 + x_3 - D_{\max} \le 0$

$g_5(x) = 3.0 - (x_2/x_3) \le 0$

$g_6(x) = \delta_p - \delta_{pm} \le 0$

$g_7(x) = \delta_p + (F_{\max} - F_p)/K + 1.05(x_1+2)x_3 - l_f \le 0$

$g_8(x) = \delta_w + (F_{\max} - F_p)/K \le 0$

The parameters used above are as follows :

$\delta_p = F_p/K$	$K = Gx_3^4/8x_1 x_2^3$
$l_f = F_{\max}/K + 1.05(x_1+2)x_3$	
$F_{\max} = 1000\, lb.$	$S = 189{,}000\, psi.$
$l_{\max} = 14.0\, inch$	$d_{\min} = 0.2\, inch$
$D_{\max} = 3.0 inch$	$F_p = 300.0\, ib.$
$\delta_{pm} = 6.0\, inch$	$\delta_{pm} = 1.25\, inch$
$G = 11.5 \times 10^6\, psi.$	

$C_f = (4(x_2/x_3) - 1)/(4(x_2/x_3) - 4) - 0.165x_3/x_2$

In this problem, the design variables are considered as integer, continuous, and discrete variables. Especially, discrete variable uses the pre-defined discrete dimensions [26].

- *Pressure Vessel Problem* (f_{pre}) : the pressure vessel problem was designed by Wu and Chow [26]. The objective function is to minimize the total cost for manufacturing the pressure vessel. The mathematical formulation is as follows :

minimize

$f_{pre}(x) = 0.6224x_1 x_3 x_4 + 1.7781 x_1 x_3^2$

$3.1661x_1 x_4 + 19.84 x_1^2 x_3$

subject to

$g_1(x) = x_1 - 0.0193x_3 \ge 0$

$g_2(x) = x_2 - 0.00954x_3 \ge 0$

$g_3(x) = \pi x_3^2 x_4 + 4/3\pi x_3^3 - 750 \times 1782 \ge 0$

$g_4(x) = -x_4 + 240 \ge 0$

$g_5(x) = x_1 - 1.1 \ge 0$

$g_6(x) = x_2 - 0.6 \ge 0$

The design variables of x_1 and x_2 are discrete values with integer multipliers of 0.0625. x_3 and x_4 are continuous values, and their side constraints are [40 inch, 80 inch] for x_3 and [20 inch, 60 inch] for x_4.

- *Reinforced Concrete Beam Problem*(f_{Rei}) : Amir and Hasegawa [1] suggested the problem of designing reinforced concrete beam. Its objective is to minimize the total cost of concrete and reinforcing steel of the beam. The mathematical formulations is as follows :

minimize $f_{Rei}(x) = 29.4x_1 + 0.6x_2 x_3$

subject to

$g_1(x) = x_1 x_3 - 7.735x_1^2/x_2 - 180 \ge 0$

$g_2(x) = -x_3/x_2 + 4 \ge 0$

In this problem, the design variables are considered as three types : integer, continuous and discrete variables : x_1 takes only certain predetermined discrete values, x_2 and x_3 take any continuous values and integer values, respectively [1].

- *Gear Train Problem* (f_{Gea}) : the gear train problem was introduced by Sandgran [19]. It is desired to produce a gear ratio as close as possible to 1/6.931. For each gear, the number of teeth must be between 14 and 40. Therefore, 〈Table 8〉 Computational results for test suit 2

	f_{Coi}			f_{Pre}		
	BFV	AFV	CPU	BFV	AFV	CPU
GA	2.0860	2.2531	3.9627	7204.5116	7360.2061	2.7384
aGA-1	2.0939	2.3572	4.1730	7224.6625	7366.1740	2.3629
aGA-2	2.0835	2.2350	5.0037	7204.9645	7367.8261	3.0219
aGA-3	2.0832	2.2534	5.3341	7206.4193	7318.3094	3.1901
aGA-FLC	2.0824	2.1366	2.1895	7243.6187	7318.7232	1.4249
aGA	2.0823	2.1056	4.3102	7200.8763	7309.4399	2.2382

	f_{Rei}			f_{Gea}		
	BFV	AFV	CPU	BFV	AFV	CPU
GA	364.8582	366.1289	0.7611	6.602E-10	5.960E-08	0.5713
aGA-1	364.8548	366.1251	0.8427	2.358E-09	4.327E-06	0.5673
aGA-2	364.8542	365.8674	0.8673	2.308E-11	2.751E-08	1.0260
aGA-3	364.8643	366.2431	1.1241	2.308E-11	2.751E-08	1.1586
aGA-FLC	364.8548	365.9753	0.8763	2.358E-09	2.493E-06	0.6225
aGA	364.8550	365.5486	0.9279	2.701E-12	7.569E-08	0.8483

the design variables are the numbers of teeth which must be integers. The mathematical formulation is expressed as follows :

minimize $f_{Gea}(x) = (1/6.931 - x_1 x_2 / x_3 x_4)^2$

subject to $12 \le x_i \le 60 \quad i = 1, 2, 3, 4$

Using the four types of test problems described above, each algorithm in <Table 5> was tested and the computational results are summarized in <Table 8>. In <Table 8>, the NGS and TGN are not appeared as the measures of performance since they are only used on the assuming that the global optimal solutions of the test problems have been known.

For each test function, the analysis is as follows.

i) f_{Coi} and f_{Pre}

For the comparison using thef_{Coi}, in terms of the BFV, all the adaptive GAs (aGA-2, aGA-3, aGA-FLC and aGA), except for aGA-1, show slightly better performances than the GA. The performance of the aGA, especially, is the best. Similar result is also shown in terms of the AFV, that is, the aGA is the best performer and the aGA-1 is the worst performer, which means that the adaptive scheme used in the aGA is more efficient in treating the rates of crossover and mutation operations than that used in the aGA-1. On the other hand, in terms of the CPU, The aGA-2 and aGA-3 is the slowest, while the aGA-FLC is the quickest.

Similar analysis results are also shown in the f_{Pre}. That is, the aGA has significantly better performances in terms of the BFV and AFV than the others, even though the search speed of the aGA is slightly slower than the aGA-FLC in the CPU.

ii) f_{Rei} and f_{Gea}

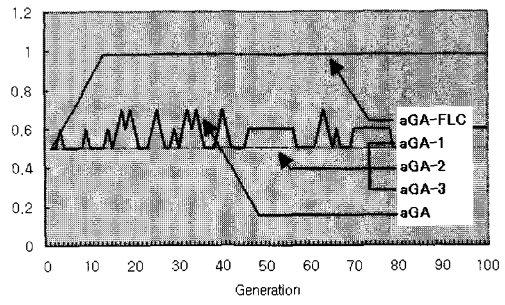
For the results of f_{Rei} in terms of BFV, the aGA-2 is the best performer. On the other hand, the performance of the aGA-3 is worse than that of the GA, even though the former has an additional adaptive scheme to regu\late P_c and P_M . Similar result is also shown in terms of AFV and CPU between the aGA-3 and GA, which implies that the adaptive scheme used in the aGA-3 does not well guide its search to much more good solutions rather than the GA does. However, in terms of the AFV, the aGA shows the best performance though its performance in BFV is slightly worse than the other adaptive algorithms (aGA-1, aGA-2, aGA-FLC).

In the comparison of BFV using f_{Gea}, the aGA is the best performer and the two aGAs (aGA-1 and aGA-FLC) are the worst. The performances of the latter are worse than that of the GA in terms of the BFV and AFV, which means that the adaptive schemes using conventional heuristic and the FLC in the aGA-1 and aGA-FLC do not be well controlling the P_c and P_M. However, the aGA shows a relatively good result than the aGA-1 and aGA-FLC in terms of the AFV, and the search speed of the aGA is quicker than those of the aGA-2 and aGA-3 in terms of the CPU.

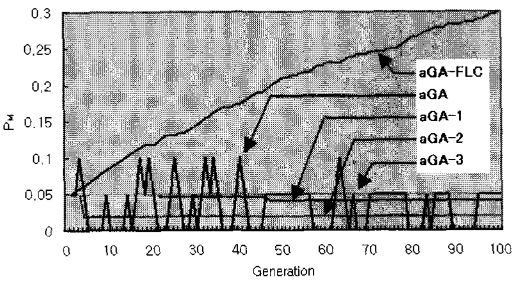
Based on the analyzed results using the test suits 1 and 2, each algorithm shows various performances. The aGA, especially, shows considerately better results in most of the measures of performance than the other competing algorithms. On the other hand, the performances of the other adaptive algorithms (aGA-1, aGA-2, aGA-3 and aGA-FLC) have significantly differences from problem to problem, and even they do not show any benefit in some measures of performance than the GA. which implies that the adaptive schemes used in the algorithms are not well controlling a balance between exploitation and exploration during the courses of their searches.

[Figure 6] and [Figure 7] respectively show the behaviors of crossover and mutation operations to prove the abilities of the adaptive schemes used in each adaptive algorithm when their searches have been reached to 100 generations in f_{Pre}.

In [Figure 6], the graph of aGA-FLC rapidly increases in initial generations, and after that, there are no changes. However, that of the aGA show various and active behaviors through all generations. In the aGA-1, aGA-2 and aGA-3, the graphs do not show any changes in all generation at all.



[Figure 6] Behaviors of crossover operation in each algorithm



[Figure 7] Behaviors of mutation operation in each algorithm

Similar situation is also shown in [Figure 7].

the aGA shows various behaviors with increasing and decreasing trends like that of the crossover operation in [Figure 6]. On the other hand, aGA-FLC shows an increasing trend continuously and the others (aGA-1, aGA-2 and aGA-3) have a little change, during all the generations.

According to the analysis using [Figure 6] and [Figure 7], we can confirm that the adaptive scheme used in the aGA is more efficient in regulating a balance between exploitation and exploration than those in the other competing algorithms, during their genetic search processes.

5. Conclusion

In genetic search process, the correct setting of a balance between exploitation and exploration is a very difficult task. In this paper, we have developed a new algorithm, the adaptive genetic algorithm (aGA), to efficiently regulate a balance between exploitation and exploration during genetic search process.

The developed aGA uses an adaptive scheme to adaptively regulate the rates of crossover and mutation operations in GA, and the key logic of its scheme is to consider the ratios of degree of dispersion resulting from the continuous two generations of GA.

In order to prove the efficiency of the aGA, various conventional adaptive GAs have been also suggested and tested using two test suits. Each algorithm, including the aGA, has been compared using various measures of performance. As a result, the performances of the conventional adaptive GAs (aGA-1, aGA-2, aGA-3 and aGA-FLC) have shown significantly various changes from problem to problem, and even on some measures of performance, they have not shown any merits when compared with the GA without any adaptive scheme. On the other hand, the proposed aGA is considerately efficient and its performance is superior to the other competing algorithms in most of the measures compared, which means that the adaptive scheme and the local search technique used in the aGA are well controlling a balance between exploitation and exploration during the genetic search process.

Reference

[1] Amir, H.M. and T. Hasegawa, Nonlinear mixed-discrete structural optimization, *Journal of Structural Engineering*, Vol.115, No.3(1989), pp.626-646.

[2] Angeline, P.J., Adaptive and self-adaptive evolutionary computations, in : M. Palaniswami, Y. Attikiouzel, R. Markc, D. Fogel, T. Fukuda, (Eds), Computational Intelligence: A Dynamic Systems Perspective, Piscataway, NJ : IEEE Press, 1995, pp.152-163.

[3] Davis, L., Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.

[4] De Jong, K.A., Analysis of the behavior of a class of genetic adaptive systems, PhD Thesis, University of Michigan (University Microfilms 1975, pp.76-9381.

[5] Espinoza, F.P., B.S. Minsker, and D.E. Goldberg, A self adaptive hybrid genetic algorithm, Proceedings on the Genetic and Evolutionary Computation Conference, San Francisco, Morgan Kaufman Publishers, 2001.

[6] Eiben, A.E., R. Hinterding, and Z. Michalewicz, Parameter control in evolutionary al-

gorithms, IEEE Transactions on Evolution Computation, Vol.3, No.2(1999), pp.124-141.

[7] Fogel, D.B. G.B. Fogel, and K. Ohkura, Multiple-vector self-adaptation in evolutionary algorithms, BioSystems, No.61(2001), pp.155-162.

[8] Gen, M. and R. Cheng, Genetic Algorithms and Engineering Design, John Wiley and Son, 1997.

[9] Grefenstette, J.J., Optimization of control parameters for genetic algorithms, IEEE Transactions on Systems, Man, and Cybernetics, No.16(1986), pp.122-128.

[10] Herrera, F., and M. Lozano, Fuzzy adaptive genetic algorithms : design, taxonomy and future directions, Soft Computing, Vol.7, No.8(2003), pp.545-562.

[11] Hoffmeister, F., and T. Bäck, Genetic algorithms and evolution strategies : similarities and differences, Proceedings of the 1st Workshop on Parallel Problem Solving from Nature (PPSN1), 1991, pp.455-471.

[12] Hong, T.P., and H.S. Wang, A dynamic mutation genetic algorithm, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, No.3(1996), pp.2000-2005.

[13] Hong, T.P., H.S. Wang, W.Y. Lin, and W.Y. Lee, Evolution of appropriate crossover and mutation operators in a genetic process, Applied Intelligence, No.16(2002), pp.7-17.

[14] Lee, C.Y., Y.S. Yun, and M. Gen, Reliability optimization design for complex systems by hybrid GA with fuzzy logic control and local search. IEICE Transaction on Fundamentals, E85-A(4) : (2002), pp.880-891.

[15] Li, B. and W. Jiang, A novel stochastic optimization algorithm. IEEE Transactions on Systems, Man, and Cybernetics-Part B : Cybernetics, Vol.30, No.1(2000), pp.193-198.

[16] Mak, K.L., Y.S. Wong, and W.W. Wang, An adaptive genetic algorithm for manufacturing cell formation, International Journal of Manufacturing Technology, No.16(2000), pp. 491-497.

[17] Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Program, Second Extended Edition, Spring-Verlag, 1994.

[18] Rabi, V. and B.S.N. Murty, P.J. Reddy, Nonequilibrium simulated annealing algorithm applied to reliability optimization of complex systems, IEEE Transactions on Reliability, Vol.46, No.2(1997), pp.233-239.

[19] Sandgren, E., Nonlinear integer and discrete programming in mechanical design optimization, ASME Journal of Mechanical Design, Vol.112, No.2(1990), pp.223-229.

[20] Srinvas, M. and L.M. Patnaik, Adaptive Probabilities of crossover and mutation in genetic algorithms, IEEE Transaction on Systems, Man and Cybernetics, Vol.24, No.4 (1994), pp.656-667.

[21] Yen, J., J.C. Liao, B.J. Lee, and D. Randolph, A hybrid approach to modeling metabolic systems using a genetic algorithm and simplex method. IEEE Transactions on Systems, Man, and Cybernetics-Part B : Cybernetics, Vol.28, No.2(1998), pp.173-191.

[22] Yun, Y.S., Genetic algorithm with fuzzy logic controller for preemptive and non-preemptive job shop scheduling problems, Computers and Industrial Engineering, Vol.43, No.3(2002), pp.623-644.

[23] Yun, Y.S. and C.U. Moon, Comparison of adaptive genetic algorithms for engineering optimization problems, International Journal

of Industrial Engineering, Vol.10, No.4(2003), pp.584-590.

[24] Wang, P.T., G.S. Wang, and Z.G. Hu, Speeding up the search process of genetic algorithm by fuzzy logic, Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing, 1997, pp.665-671.

[25] Wu, Q.H., Y.J. Cao, and J.Y. Wen, Optimal reactive power dispatch using an adaptive genetic algorithm, Electrical Power and Energy Systems, Vol.20, No.8(1998), pp.563-569.

[26] Wu, S.J., and P.T. Chow, Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization, Engineering Optimization, No.24(1995), pp.137-159.

[27] Shuguang, Z. and J. Licheng, Multi-objective evolutionary design and knowledge discovery of logic circuits based on an adaptive genetic algorithm, Genetic Programming and Evolvable Machines, No.7(2006), pp.195-210.