# A Genetic Algorithm with Ageing Chromosomes

# 나이를 먹는 염색체를 갖는 유전자 알고리즘

Sung Hoon Jung*
정 성 훈*

ABSTRACT

This paper proposes a modified GA whose individuals have their own ages. Thus, a chromosome will die only when the age becomes zero, as a result, the population size of this method increases according to the generations. This helps a GA to preserve the good characteristics of a few chromosomes during several generations if the ages are evaluated with fitness values. As a result, the performance of the method is better than that of existing ones. A multi-modal function optimization problem is employed to simulate the performance of this method. To show the effectiveness of ageing paradigm, three ageing evaluation methods are introduced. A paper whose idea is similar to that of ours have been published in a conference. We also experimented a method that showed the best performance in the paper. Original simple GA was also experimented and the performance is compared with others. However, the performance of the previous method shows worse than that of our methods in some aspects because the previous method didn't take the fitness value into account in the selection process.

요 약

본 논문에서는 각각의 유전자 개체가 그 자신의 나이를 갖는 수정된 유전자 알고리즘을 제안한다. 그래서 하나의 개체는 그 자신의 나이가 0이 되었을때 만이 사라지게 되며, 결국 개체들의 수가 동적으로 증가할 것이다. 이러한 방법은 좋은 특성을 갖는 유전자들을 그들의 적합도에 상응하게 매겨진 나이에 따라서 몇 세대에 걸쳐 살아있게 함으로서 그들의 좋은 형질을 좀더 보존할수 있는 능력을 유전자 알고리즘이 갖추게 만든다. 그러므로, 이러한 방법은 기존의 방법보다 더욱 좋은 성능을 내게된다. 우리는 이 방법을 실험하기 위하여 함수최적화 문제를 이용하여 시뮬레이션하였다. 제안한 방법의 효율성을 보이기 위하여, 세가지의 나이를 먹는 방식을 제안하였다. 본 논문의 아이디어와 유사한 연구논문이 있었는데, 이 연구에서 성능이 가장 좋은 방식을 채택하여 실험하였다. 또한 원래의 간단한 유전자 알고리즘도 실험을 하여 성능을 비교 하였다. 그러나, 기존의 방식은 나이를 산정할때 적합도를 고려하지 않음으로서 본 논문에서 제안한 방식보다 몇몇 측면에서 성능이 좋지 않았다.

## I. Introduction

Genetic Algorithms(GAs) methods based on the genetic processes of biological organisms are adaptive methods for solving search and optimization pro-

*한성대학교 정보전산학부

blems [1, 2, 3, 4]. GAs are robust, and can be applied to a wide range of problem areas that are difficult to solve. The solutions of GAs are generally good at finding *acceptably good* solutions *acceptably quickly*. A GA operates through a simple cycle of stages: (1) creation of a population of strings, (2) evaluation of each string, (3) selection of the best strings, and (4) genetic manipulation to create a new population of strings.

In existing GAs, the good characteristics of genetically manipulated parents are sometimes destroyed in the next generation. This is because such parents are permanently deleted from the population, even if the fitness of the generated offsprings are not better than that of the parents. Moreover, a currently bad chromosome with a high latent faculty may generate good offsprings in some next generation. From these observations, we think that there is only one chance for parents to generate offsprings. That is, all chromosomes in a population have a few chances to create offsprings for a few generations in proportion to their fitness.

This paper proposes a genetic algorithm having ageing chromosomes. Each individual in our genetic algorithm lives for a few generations. Their lifetimes correspond to the value of its age that is derived from the fitness. In this paper, we will show that this method increases two probabilities: one is that good characteristics of a few chromosomes can be preserved, another is that good characteristics can be revealed from currently bad chromosomes.

We simulated our method with a multi-modal function optimization problem. In this simulation, three equations for calculating the age of each individual are used. A similar paper to ours have been published in a conference [5]. In the paper, the authors took lifetime for each chromosome. They, however, didn't take the fitness of a chromosomes into account in the selection process. We experimented a method that showed the best performance in the paper. Original simple GA was also experimented and the performance is compared with others. Simulation results show that our method outperforms the other two methods in some aspects.

This paper is organized as follows. Section 2 provides a brief description of genetic algorithms. Our genetic algorithm and a similar algorithm are described in section 3. Simulation and discussion is given in section 4. This paper is concluded with section 5.

## Ⅱ. Brief Review of Genetic Algorithms

GAs are adaptive methods for solving search and optimization problems [1, 2, 3, 4]. They are based on the genetic processes of biological organisms. Natural populations over many generations evolve through natural selection and survival of the fittest. GAs simulate those processes in natural populations which are essential to evolution. It has been shown that these GAs can well be applied to real world problems if they are suitably encoded. The combination of good characteristics from different ancestors can sometimes produce a superfit offspring, whose fitness is greater than that of either of the parent. In this way, species evolve to become more and more well suited to their environment.

The power of GAs comes from the fact that the technique is robust, and that can deal successfully with a wide range of problem areas, including those which are difficult for other methods to solve. GAs are not guaranteed to find the global optimum solution to a problem, but they are generally good at finding *acceptably good* solutions to problems *acceptably quickly*.

A GA operates through a simple cycle of stages: (1) creation of a population of strings, (2) evaluation of each string, (3) selection of best strings, and (4) genetic manipulation to create a new population of strings. Algorithm 1 shows the structure of a simple genetic algorithm.

---

**Algorithm 1** Simple Genetic Algorithm

//t: time//

//$P$: populations//

```
1  t ← 0
2  initialize P(t)
3  evaluate P(t)
4  While (not termination-condition)
5  do
6     t ← t + 1
7     select P(t) from P(t − 1)
8     recombine P(t)
9     evaluate P(t)
10 end
```

Before a GA can be applied to a problem, the problem must be coded. Then, the specific number of initial individuals are generated according to the coding method. These coded genes are evaluated to measure the fitness of the solution. To make a next generation, selection and recombination are necessary. This procedure does not stop until satisfiable solutions are obtained. More detailed descriptions about GAs are available in [1, 2].

## Ⅲ. A Genetic Algorithm with Ageing Chromosomes

The number of individuals in existing GAs is fixed because their parents generate the same number of individuals. Even if good characteristics of chromosomes in offsprings are spread throughout the population, high latent faculties of a few chromosomes - they have currently low values of fitness - may die out. This is because roulette wheel selection provides only a chance to keep their characteristics in the next generation. Especially, when a few individuals have very low values of fitness, they have no chance to preserve their characteristics. Moreover, there are no theorems for deciding the initial number of individuals even if the initial number of individuals affects the performance of GAs [6, 1].

This paper proposes a genetic algorithm with ageing chromosomes from these observation. In this method, all individuals have their own ages which are given from fitness measures. Each individual remains alive for a few generations according to the value of its age. We take three methods in deciding the age. These methods are simulated and compared to each other. In our GA with ageing chromosomes, good characteristics of chromosomes will be conserved throughout the solution process. Algorithm 2 shows our modified GA, termed an *Ageing GA* (*AGA*).

**Algorithm 2** Ageing Genetic Algorithm

```
   // t : time //
   // PP : populations pool //
   // SP : selected populations //
1  t ← 0
2  initialize PP and initial populations
3  evaluate each chromosome in PP
4  decide the age of each chromosome
5  While (not termination-condition)
6  do
7     t ← t + 1
8     select populations from PP
9     recombine SP
10    evaluate SP
11    determine the ages of new chromosomes
12    insert the new chromosomes into PP
13    subtract 1 from the values of ages of all chromosomes
14    remove chromosomes having zero age
15 end
```

In our AGA, the age of a chromosome is determined from three strategies. First method selects the age of each individual from a random number generator with an allowable maximum value given by the designer. Second method is the same as the first one except the fact that the allowable maximum random number is calculated from the fitness measure. Thirdly, the calculated value in the second method is taken as the ages not the allowable maximum number.

$$A_i = 1 + random(A_{max}) \quad (1)$$

$$A_i = 1 + random(A_{max} * f_i / f_{max}) \quad (2)$$

$$A_i = 1 + A_{max} * f_i / f_{max} \quad (3)$$

Equation 1-3 shows the equations of the three methods *random*($x$) produces a random number from zero to $x$, $A_{max}$ is the maximum allowable age, $f_i$ is the fitness of i'th chromosomes, and $f_{max}$ is the current maximum fitness value. The three methods will be called *random ageing method*($RAM$), *calculated random ageing method* ($CRAM$), *and calculated ageing method*($CAM$), respectively.

Arabas *et al.* presented a method-termed GAVaPS (Genetic Algorithm with Varying Population Size)-similar to our method in a paper [5]. The major idea of the paper is the same as that of this paper. They, however, didn't take the fitness of a chromosomes into account in the selection process. That is, their method randomly selects the parents without considering the fitness of chromosomes. The paper also introduced a *reproduction ratio* that was given as: $AuxPopSize(t) = [PopSize(t) * \rho]$ where $AuxPopSize(t)$ is newly generated offspring size, $PopSize(t)$ is current population size, and $\rho$ is a reproduction ratio. Their experiments was brought into focus on the performance of their method according to the reproduction ratio. Three ageing strategies, *proportional allocation, linear allocation, bi-linear allocation,* was provided and experimented. The three strategies are as follows.

$$A_i = min(MinLT + \gamma \frac{f_i}{AvgFit},\ MaxLT) \quad (4)$$

$$A_i = MinLT + 2\gamma \frac{f_i - AbsFitMin}{AbsFitMax - AbsFitMin} \quad (5)$$

$$A_i = MinLT + \gamma \frac{f_i - MinFit}{AvgFit - MinFit} \quad if\ AvgFit \geq f_i \quad (6)$$

$$A_i = \frac{1}{2}(MinLT + MaxLT) + \gamma \frac{f_i - AvgFit}{MaxFit - AvgFit}$$

$$if\ AvgFit < f_i \quad (7)$$

where $MinLT$ and $MaxLT$ are values of minimal and maximal allowable lifetime, respectively; $AvgFit$, $MaxFit$ and $MinFit$ represents average, maximal and minimal fitness values, respectively; $AbsFitMin$ and $AbsFitMax$ stand for maximal and minimal fitness values found so far; $\gamma$ is $\frac{1}{2}(MaxLT - MinLT)$. Four multi-modal functions were employed to test their methods. Experimental results of the paper showed that the linear allocation method found the best solution among the methods, but the function evaluation cost of the method is the highest.

## Ⅳ. Simulation

To illustrate the performance of the AGA, a multi-modal function that is an example of the paper [5] is used.

$$f(x) = -x\sin(10\pi x) + 1, \quad where\ -2.0 \leq x \leq 1.0 \quad (8)$$

The first step in optimizing the function over the interval [0-255] is to encode the parameter set $x$, that is, an eight-digit binary string 00000000-11111111. Next, $N$ initial chromosomes are generated using a random number generator. These $N$ chromosomes are inserted into the populations pool. Such individuals in the $PP$ are evaluated by fitness measure and selected to recombine new individuals. Newly generated individuals are evaluated their ages, and inserted into the $PP$. The GAVaPS algorithm and simple genetic algorithm were also implemented to compare the performance with our methods. We experimented our three ageing methods, the linear method that was shown the best performance in [5], and original simple GA. Three parameters-the number of initial individuals, the methods of ageing, and the allowable maximum ages-were taken to experiment. Table 1 shows the values of simulating parameters.

This paper takes five performance measures: 1) the first generation number for finding the optimum sol-

Table 1. Simulation Parameters

| | |
|---|---|
| the number of initial individuals | 10, 30, 50, 80 |
| the methods of ageing | 1, 2, 3, 5 |
| the maximum ages | 4, 8, 12, 16, 20 |

ution, 2) the variance of value of maximum fitness after reaching the optimum solution, 3) the average number of chromosomes, 4) the average age of the generated chromosomes in each generation, 5) the first measure per the third measure. Fifth performance measure indicates that the searching ability of a method under same number of chromosomes. Thus, this measure is the most important. The variance, termed "nipple" in this paper, is given as follows.

$$V = \frac{\sum_{l=find\_best}^{max\_generations} (f_l^{max} - MaxFit)^2}{N} \quad (9)$$

where the *find_best* means the first generation number of finding the optimum solution and the *max_generations* indicates the final generation number that is given by human as a parameter; where $f_l^{max}$ is the maximum fitness value of l'th generations, *MaxFit* is the best solution, and $N$ is the number of added nipple terms.

The performance of GAs is based on the operators whose operations depend on the probability of each operator. Also, the parents are selected randomly from the population using a scheme which favors the more fit individuals. These facts indicate that the performance of a GA is probably better than that of others depending on the operations. Thus, we take a statistic manner to evaluate the characteristics of each simulation method. That is, we take average values of the parameters. For example, if we pay attention to the variation of the first performance measure in comparison with the number of initial individuals, then the results of first measure in all maximum ages are taken to calculate the average value of the first measure.

Figure 1-5 shows the results of all performance measures. The maximum number of generations in the experiments is set to 300. Thus all experiments are finished when the the number of generations becomes

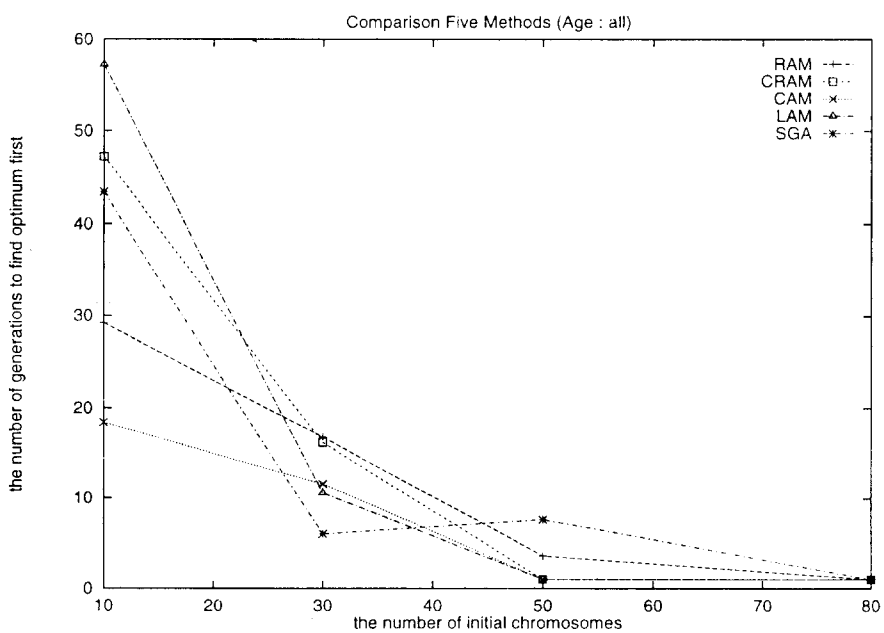


Figure 1. First generation number to find an optimum solution in each methods

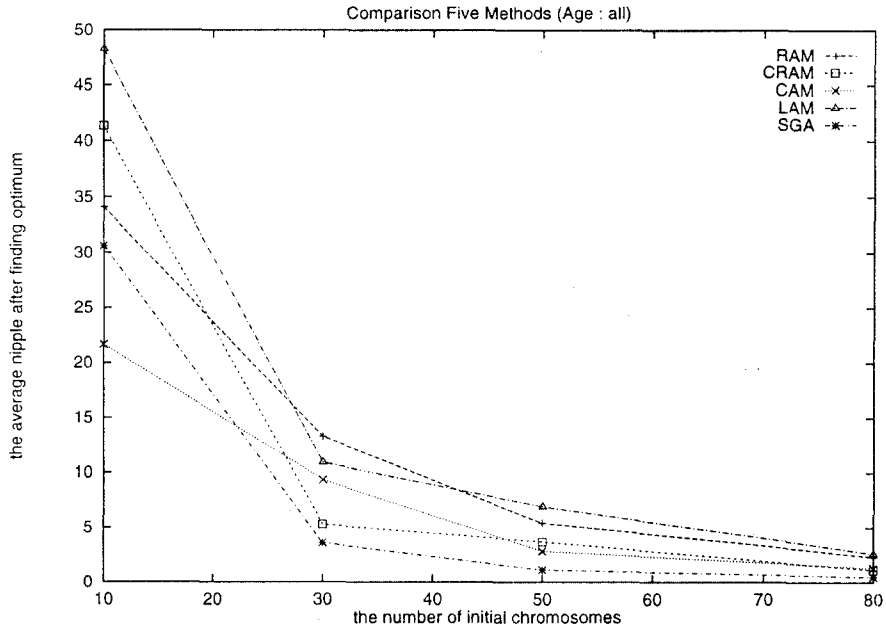


Figure 2. Variance after reaching an optimum solution

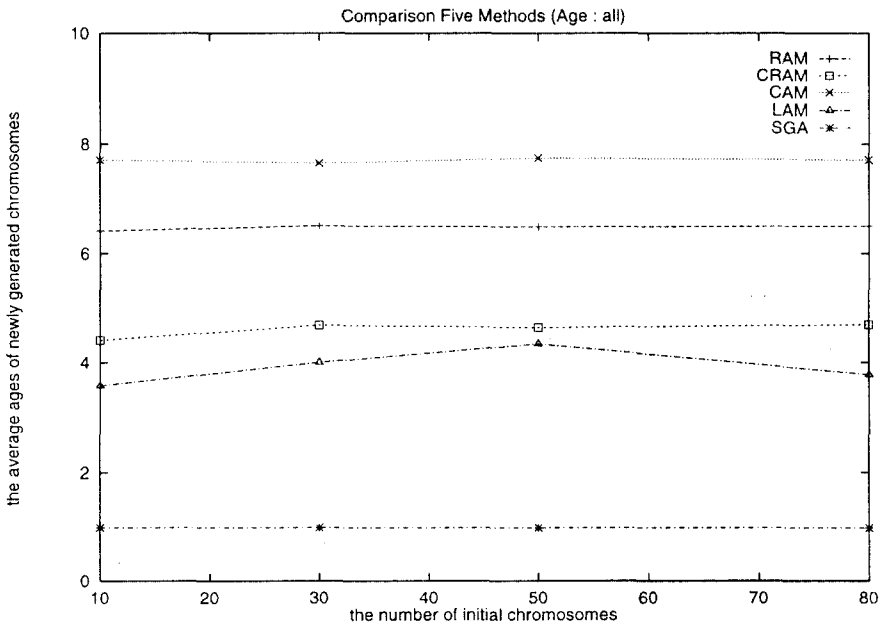


Figure 3. Average ages in each generation

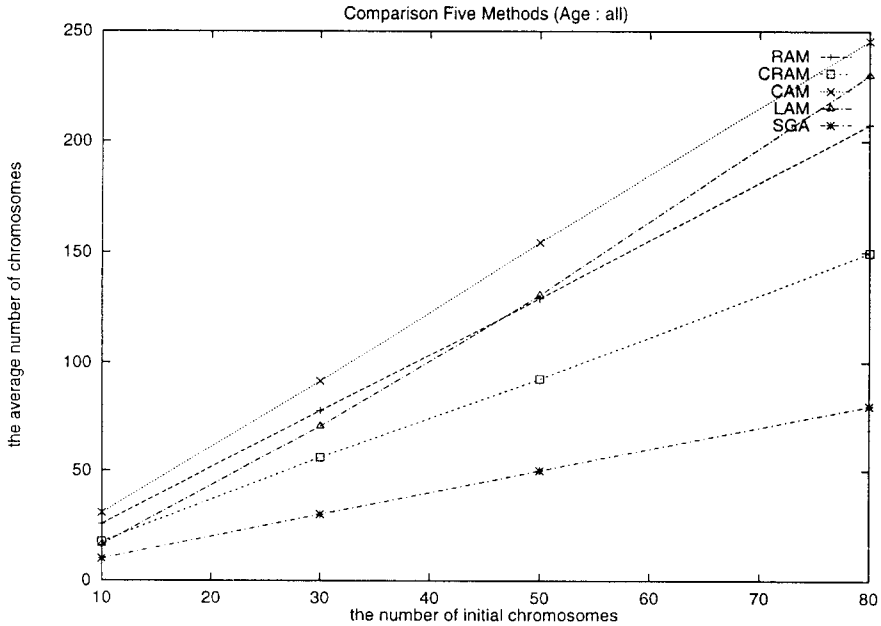


Figure 4. Average number of chromosomes

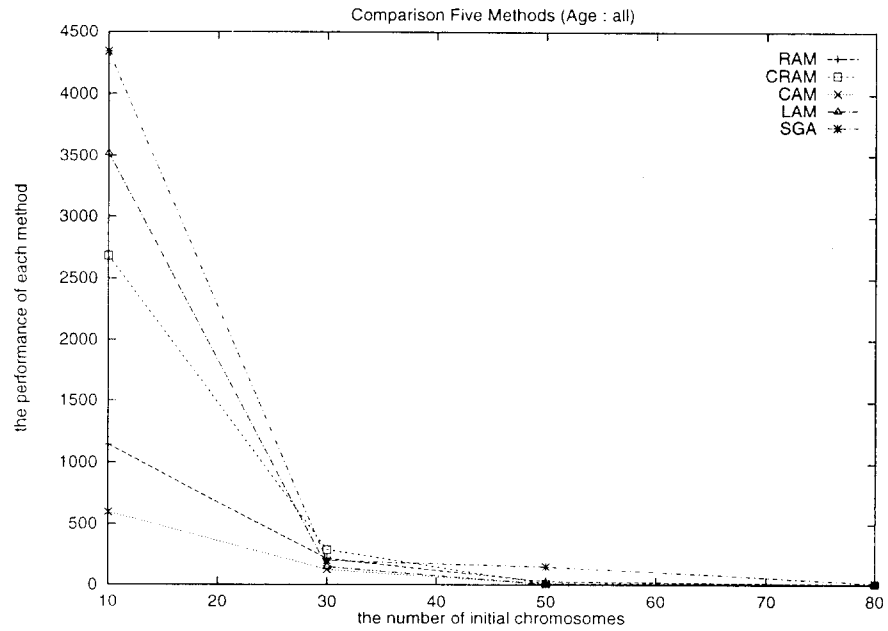


Figure 5. Performance of each method

300. As shown in Figure 1, CAM outperforms the four other methods when the number of initial chromosomes is 10. According as the number of initial chromosomes increases, however, the performance of SGA and LAM are greatly enhanced than that of other methods. The performances of LAM are good where the number of initial chromosomes is greater than about 30. This result may be due to the fact that although the age of a chromosome has better proportion to its fitness value, the effect is not large when the the number of initial chromosomes are larger than about 30. Selecting the parents with their fitness shows better in case of only small initial chromosomes as shown in figure 1. As mentioned in section 3, LAM randomly selects the parents without considering their fitness. Thus, the fitness of chromosomes in the method affects only the calculation of lifetime of each offsprings. That is, where the initial chromosomes are greater than 30 the performance of CAM and LAM is nearly same each other. The performance of RAM shows considerably bad performance where the number of initial chromosomes is less than 30. However, RAM shows nearly same performance as the others where the number of initial chromosomes is greater than 80. SGA shows medium performance, however, the increasing rate of performance is lower than that of others. Above 30 initial chromosomes, the SGA shows the good performance and the difference of performance among the methods are reduced, this indicates that the ageing methods do not effect the performance where the number of initial chromosomes are many. Figure 2 shows the variance of the maximum fitness of each generations after finding optimal solutions. This measure is an important one in a point of view that this measure indicates how much the method can keep the good fitness of chromosomes. In figure, the result of each method is very similar one another. However, the SGA shows the lowest variance where the number of initial chromosomes are greater than about 30. This may be because the effect of good fitness of alive chromosomes decreases when the initial chromosomes is large. The CAM and RAM shows relatively good performance in this viewpoint. The average age of each method is shown in figure 3. We took five maximum ages-4, 8, 12, 16, 20 as shown in table 1. We averaged the age of all generated chromosomes in each experiment. As shown in figure, the average age of CAM is the largest. It is very natural result that the average age of SGA is 1. Figure 4 shows the average number of chromosomes in each method. The average number of chromosomes in SGA is naturally the same as the number of initial chromosomes. The other methods, however, increase the average number of chromosomes although the initial chromosomes are constant. CAM shows the largest average number of chromosomes. Generally, according to increasing the number of chromosomes, the performance of GA is also increased. Thus, we take a new performance factor as: $G_f/A^i * 1000$, where the $G_f$ is the first number of generation of finding optimum and $A^i$ the average number of chromosome in each method. Figure 5 shows the performance. The CAM shows the best performance and the RAM and CRAM present relatively good performance. The performance of LAM is better than that of SGA, however, it is not better than that of RAM, CRAM, and CAM. When initial number of chromosomes are large, the difference of performance among the methods is reduced. This indicates that the ageing method greatly affects in the case that the number of initial chromosomes are small. Simulation results indicate that the CAM method is the best one to find the final optimum solution even when the performance is divided by the average number of chromosomes.

AGA is an extended method of the original GA. This method provides a designer with a general and flexible environment for designing the structure of a GA. This is because in an existing environment, the fitness of a chromosome affects only the selection, but AGA can affect the decision of age.

## V. Conclusion

A modified GA having ageing chromosomes is designed and simulated. This method provides a designer with another dimension of operators to devise the operations of a GA. The performance of AGA can be varied according to versatile ageing methods. A designer can devise his own ageing method to improve the performance. Our method is similar to the GAVaPS one. Simulation results of a multi-modal function optimization, however, show that the CAM method outperforms four other methods. This is caused by a fact that the age of chromosomes had better proportion to their fitness than random. The performance of LAM is worse than that of CAM because the method didn't take the fitness value into account in the selection process. According to application areas, the ageing methods can be modified to fit in the applications.

## References


1. M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," *IEEE Computer Magazine*, pp. 17-26, June 1994.
2. J. L. R. Filho and P. C. Treleaven, "Genetic-Algorithm Programming Environments," *IEEE Computer Magazine*, pp. 28-43, June 1994.
3. G. A. Vignaux and Z. Michalewicz, "A Genetic Algorithm for the Linear Transportation Problem," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 21, pp. 445-452, MARCH/APRIL 1992.
4. C. L. Karr and E. J. Gentry, "Fuzzy Control of pH Using Genetic Algorithms," *IEEE Trans. on Fuzzy Systems*, vol. 1, pp. 46-53, Jan. 1993.
5. A. J., M. Z., and M. J., "GAVaPS-a Genetic Algorithm with Varying Population Size," *Proceedings of the Evolutionary Computation Conference, part of the IEEE World Congress on Computational Intelligence*, June 1994. Orlando.
6. M. Srinivas and L. M. Patnaik, "Adaptive Probabilities of Grossover and Mutation in Genetic Algorithms," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24, pp. 656-667, Apr. 1994.



정 성 훈(Sung Hoon Jung) 정회원

1984년 3월~1988년 2월 : 한양대학교 전자공학과 공학사

1989년 3월~1991년 2월 : 한국과학기술원 전기 및 전자공학과 공학석사

1991년 3월~1995년 2월 : 한국과학기술원 전기 및 전자공학과 공학박사

1995년 3월~1996년 2월 : 한국과학기술원 전기 및 전자공학과 위촉연구원

1996년 3월~현재 : 한성대학교 정보전산학부 전임강사