以新的運算子來改良基因遺傳法則
Improving the genetic algorithms by new operators

周智勳* 陳洲男
Chih-Hsun Chou Jou-Nan Chen

*中華大學資訊工程系
Department of computer science, Chung-Hua University
chc@chu.edu.tw

摘要

雖然基因法則已展現了它在搜尋及最佳化方面的效能，增進其準確度及收斂速度仍是必須的研究課題。許多的研究針對基因法則的操作子及其相對應的參數做了不少的探討，而本論文則針對基因法則提出新的操作子稱之爲「萃取」。它能將一群較好的染色體中萃取出好的基因，以遺傳給下一代，並破壞不好的染色體中不好的基因。另外我們亦提出兩個新的初始值設定方法。實驗證明我們提出的這些方法對基因法則有很明顯的改良作用。

關鍵字：操作子，基因萃取，初始值設定

Abstract

Though genetic algorithms have shown their power in search and optimization, improving in solution accuracy and convergence speed are still the necessary researches. Many researchers on genetic algorithms lay great emphasis on the genetic operators or the parameter adjustment. In this paper, however, we propose a new operator called extraction. The common alleles of several best individuals of the population are extracted as good genes (the alleles' values are the same), and reserved with a probability, while the other genes are destroyed with another probability. We also propose two advanced population initialization methods that force the individuals' distribution to be uniform in the search space. Experimental results show that these methods evidently improve the performance than that of the simple genetic algorithms.

Keywords: operator, extraction, initialization

1. Introduction

Genetic algorithms (GAs) are robust search and optimization algorithms based on natural selection in environments and natural genetics in biology. That is, they are based on the natural evolution according to the principle of survival of the fittest (natural selection). GAs are first proposed by Holland in 1970s. Theoretical developments by Holland [12] and DeJong [7] have laid the foundations of GAs.

GAs have been successfully applied to many fields of optimization, machine learning, adaptive system design, neural networks, image processing, pattern recognition, biological simulation, and others [6, 10]. GAs have been theoretically and empirically proven to have the efficiency of robust search in complex spaces [10].

In many cases, GAs are better than traditional methods in optimal search, but they have their own problems from the nature. These problems are described as follows:

1. *Premature convergence*

It prevents GAs from finding the optimal solution, and causes GAs to converge to suboptimal or undesired point. The selection, crossover and mutation operators may cause this result. The main reason is that the superior individual appearing in the early evolution stage, and it dominating the entire population quickly. In biology, if all individuals of a population are the same, and their fitnesses are low, then they will be destroyed fully by the natural environment. Crossover and mutation also may destroy the good individuals of the population.

2. *Convergence speed*

If the similarity of the population is too high, then the convergence speed is low. If the convergence speed is low, and the fitness of the population is low, then it will cause a poor evolution.

3. *Execution time*

The execution speed of a GA is slowed if there are mass computation of evaluating fitnesses and operations of genetic operators during the evolution.

In this paper, we propose some novel valuable methods including gene extraction and advanced initializations. These methods have better performances than simple genetic algorithms and other methods, especially the extraction operator. The extraction operator alters the structure of the individual. It analyzes and extracts the good genes and the bad genes, and

reserve the good genes and destroy the bad ones. The **uniform initialization** method modifies the population initialization. It makes each initial individual locating on the diagonal linear subspaces of the search space uniformly. The **unbiased initialization** method is also used to modify the population initialization. It makes the population initialization unbiased. That is, the alleles (the genes at the same position) contain the same number of ones and zeros. All these methods are described in section 3 in details.

This paper is organized as follows. In section 2, the terminologies of GAs and some previous improved researches on GAs are examined. The ideas and algorithms of our improved methods are proposed in section 3. Experimental results are showed in section 4. In section 5, some conclusions are given.

2. Preliminary

Before presenting the proposed method, some terminologies are stated.

2.1 Terminology

1. *Gene*
   Abit is usually used to represent a gene in GAs. One or some genes can represent a feature in GAs.
2. *Chromosome/individual*
   A binary string that represents a solution of the problem in GAs. It may contain several variables depending on the problems.
3. *Locus*
   The position of the gene in a chromosome.
4. *Allele*
   The gene at the same locus.
5. *Schema*
   The template of some chromosomes. For example, the schema H = 1001**01 contains four chromosomes such as 1 0 0 1 0 0 0 1, 1 0 0 1 0 0 1 1, 1 0 0 1 0 1 0 1 and 1 0 0 1 0 1 1 1.
6. *Population*
   A set of individuals $P=\{S_1, S_2,..., S_n\}$.

2.2 About the improvements on Genetic Algorithms

Many researchers on GAs attempt to improve the performance of GAs. They lay great emphasis on the improvements of the mechanisms and rates of genetic operators, and they have proposed the methods of improved operators [1, 3, 4, 6, 8, 15, 17] and adaptive operator rates [5, 6, 9, 17]. These adaptive methods include adaptive crossover and mutation rates, varying population size and varying encoding length. Other improving methods such as real number encoding and multi-populations are also available in [6, 13, 14, 18].

3. Gene Extraction and Advanced Initializations

The main problems of simple genetic algorithms are the premature convergence, the convergence speed and the execution speed. To overcome these problems, in this section, we create an artificial genetic operator to reserve the good individuals and reduce or replace the bad ones, and to increase the average and maximum fitnesses of the population.

3.1 Extraction

Genetic algorithms have the ability of locating the area of the optimal or acceptable solution with the evolution. Even simple genetic algorithms can find an acceptable solution easily, and sometimes provide a better performance than other modified methods. New operators can be combined with the simple genetic algorithm to enjoy both benefits. So we have the idea of creating a new operator to make the bad individuals destroyed and the good ones reserved. To determine which individuals are good and which are bad, we adapt the average fitness of the population as the threshold. That is, if the individuals' fitnesses are bigger than or equal to the average, then they are good individuals, else bad ones. Similarly, in an individual, the bad genes can be destroyed and the good ones should be reserved. But which are **good or bad genes** of the individual? Generally, better individuals have more or important desired genes (the solution schema) since they have higher fitnesses than others. The above ideas lead to our new operator that is based on the following statements:

1. For any problem with single or multi solutions, it has the solution schema (the schema of optimal solutions). If an individual matches more genes of the solution schema, then the individual has higher probability to be the optimal solution.
2. In statement 1, it is necessary to consider the gene weight (the importance of the gene). The genes that are more important can be found easier since they make higher fitnesses than the others.
3. The better individuals of the population have higher probability to contain the desired genes (the solution schema) since they have higher fitnesses than the others.
4. In statement 3, the common alleles of those better individuals may be the desired genes (the solution schema).

Basing on the above statements, we conclude that the common alleles of the better individuals play an important role in finding the solution schema. Hence, at every generation, the common alleles are extracted as the desired genes. These common alleles are reserved and the others are replaced, both with random values.

It is because that the common alleles of the better individuals have higher probabilities to match the solution schema, the common alleles are reserved with a probability, and the others are destroyed with another probability. If the locus of a gene in an individual is equal to the locus of a good gene, then the good gene is put into the locus of this individual with a higher probability, else the gene at the locus is replaced with random bit with a lower probability. We call this operator as **extraction**.

For example, assuming that the encoding length is 8, and the five best individuals are as list in figure 3.1. Consider the locus 1, since all of the genes are "1", the alleles are the same, so locus 1 should be a good gene and has a value 1.

We now use figure 3.1 to describe the extraction operator step by step as follows:

1) Sort the individuals according to their fitnesses after a generation.
2) Make two arrays *A [1..Len], B[1..Len]*, where *Len* is the encoding length, where
   A indicates that whether the alleles are common ones or not, in which 1 represents yes, and 0 represents no;
   B indicates the values of the common alleles (1 or 0).
3) Initialize *A* as a zero array (no common alleles at the beginning).
4) For alleles at locus *i* of the five best individuals do step 5 and 6.
5) Compute the number of '1' of the alleles at locus *i* of the five best individuals. Let the number be *n*.
6) If $n = 5$ (all the five alleles are 1's), then $A[i] = 1$ and $B[i] = 1$, else $A[i] = 0$ and $B[i]$ neither equal to 1 nor to 0.
   If $n = 0$ (all the five alleles are 0's), then $A[i] \leftarrow 1$ and $B[i] \leftarrow 0$, else $A[i] \leftarrow 0$, and $B[i]$ neither equal to 1 nor to 0.
7) For alleles at locus *i* of the remaining individuals do step 8.
8) If $A[i] = 1$, then the *i*th gene of each of these individual equals to $B[i]$ with probability 0.6.
   If $A[i]=0$, then the *i*th gene of each individual equals to a random bit with probability 0.6.

Best individual 1	1 0 1 0 1 0 1 0
Best individual 2	1 1 1 0 1 0 0 0
Best individual 3	1 0 1 0 0 0 0 0
Best individual 4	1 0 1 0 1 1 1 0
Best individual 5	1 1 1 0 0 0 1 0
Array *A*	1 0 1 1 0 0 0 1
Array *B*	1 1 0 0

Figure 3.1 Extracting the good genes

9) Recompute the fitnesses of these remaining individuals.
10) Enter next generation.

There are four parameters in the extraction operator that influence its performance.

1. *Reserve rate*
   The probability of reserving the good genes. Our experimental results show that the range from 0.4 to 0.8 gives a good performance. So, the default value is 0.6.
2. *Replacement rate*
   The probability of destroying the bad genes. This is the probability of replacing the bad genes with random values. Our experimental results show that the range from 0.4 to 0.8 gives a good performance. Hence, the default value is set as 0.6.
3. *The number of best individuals for comparison*
   The number of best individuals that are used to extract good genes. We find that the range from 3 to 10 gives a good performance, and our default value is 5.
4. *Interval of executing extraction*
   This is the interval number of generations that the extraction operator operates. Our experimental results show that the range from 1 to 5 gives a good performance, and we set the default value as 1.

3.2 Uniform Initialization

Generally speaking, the initialization of the individuals of the population should be as uniform distributed as possible. But it is hard to achieve because the initialization is random. The uniform initialization, however, gives the GAs a better performance because there is an individual in each area of the search space. So, unless the property of the problem is known, and a special initialization is used, otherwise in general problems, we force that there is an individual locating in each subspace of the search space. That is, the search space is splitted into subspaces of population size to make each subspace contain an initial individual.

It is practicable to do so for one-dimensional problems but is not so easy for multi-dimensional ones. For example, consider a two-variable problem, assume that the population size is 30, then the search space is splitted into 30 subspaces. But for two-dimensional space, it can only be splitted into subspaces such as 1×30, 2×15, 3×10 and 5×6 types. And for five variables, it can only be splited into $1\times1\times2\times3\times5$ type subspaces. Hence, uniform initialization ($n\times n$, $n\times n\times n\times n\times n$) is not so easy to obtain.

To overcome this problem, we find that for many problems, the diagonal linear subspaces of the search space contain important information. In our eleven

popular test problems, there are nine problems whose optimal solutions are located in the diagonal linear subspaces. For the other two problems, there are still high fitness solutions in the diagonal linear subspaces. According to the above observations, in our method, each dimension is splitted into intervals with an amount of the population size. So it will forms Pop^{Param} subspaces (*Pop* is the population size, *Param* is the number of variables or dimensions). Only the diagonal linear subspaces are extracted. For example, assuming that there are four individuals in the population (*Pop* = 4), and the search space is two-dimensional (*Param* = 2). The origin initialization is illustrated in figure 3.2(1). The first proposed uniform initialization and the linear diagonal uniform initialization are demonstrated in figure 3.2(2) and figure 3.2(3), respectively. In the figures, gray areas indicate that there is an individual in it, whereas the blank areas are not.

The linear diagonal uniform initialization (in short **uniform initialization**) is described as follows:

Step 1. Compute the interval size used to split each dimension

$$Interval = \frac{\text{space size}}{\text{population size}}$$

Step 2. Locate the initial values

The value of each parameter of individual i = a random number I between $interval*(i-1)$ and $interval*(i)$, where * means multiplication.

Step 3. Convert those initial value I of the individuals into the binary string of a parameter.

Step 4. Repeat step 2 and 3 for each individual.

3.3 Unbiased initialization

Unbias means that the count of "1" in the alleles is equal to the count of "0". If the alleles at some locus are the same, then it is impossible to alter the genes at this locus by crossover. Hence, initialization with unbiasing should be benefit for the crossover operator. In most cases, however, the initial population is usually biased, so we propose an algorithm that can force it to be unbiased. The algorithm is as follows:

Step 1. Let the $(2*i-1)$th individual be "111...1" and the $(2*i)$th individual be "000...0", where i is from 1 to the population size:

Each gene of every individual i = the Boolean operation (i AND 1).

Step 2. Select two alleles at locus j randomly to exchange their values.

Step 3. Repeat step 2 population-size times.

Step 4. Repeat step 2 and step 3 for each locus (j is from 1 to string length).

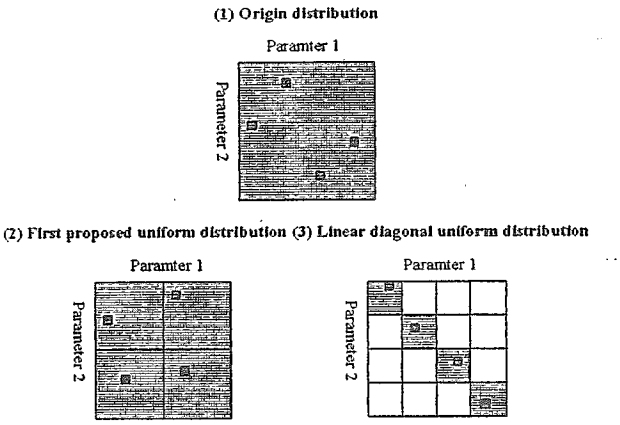

Figure 3.2 Demonstration of the uniform initialization

Individual 1	1	1	1	1	1↔2	0	1	1	1
Individual 2	0	0	0	0	1↔3	1	0	0	0
Individual 3	1	1	1	1	2↔4	0	1	1	1
Individual 4	0	0	0	0	1↔2	1	0	0	0

Figure 3.3 An example for unbiased initialization

For example of step 2, consider the four individuals in figure 3.3. If the first genes of individual 1 and 2, and then 1 and 3, and then 2 and 4, and finally 1 and 2 are exchanged, then the four new individuals are as shown in the right of the figure which is unbiased.

4. Experiments and Results

In this section, the performance measures and test problems are introduced before displaying the experimental results. In the experiments, we compare the performances and execution times of our improved methods with the simple genetic algorithm's. Besides, we also compare the performances of the extraction operator and other improved methods with various parameters setting.

4.1 Performance Measures

To know how good the performances of genetic algorithms are, we must formulate some measures. Because GAs involve randomness, each method is run 50 times. The performance measures are as follow:

1. *Average curve*

f_j^i : the fitness of the jth individual at generation i.

f_{avg}^i : the average of all individuals' fitnesses at generation i.

2. *Maximum curve*

f_{max}^i : the maximum fitness of all individuals at

generation i.

3. *Average fitness*

f_{avg}: the average of average fitnesses of all generations.

$$f_{avg} = \frac{\sum_{i=1}^{m} f_{avg}^{i}}{m},$$

where m is the number of evolution generations.

4. *Maximum fitness*

f_{max}: the average of maximum fitness of all generations.

$$f_{max} = \frac{\sum_{i=1}^{m} f_{max}^{i}}{m},$$

where m is the number of evolution generations.

5. *Best fitness*

f_{best} indicates the best of maximum fitnesses of all generations. The individual with best fitness is the solution. The individual is the precise solution if its fitness is equal to 1.0 after normalizing. It is impressed on the average and maximum fitnesses typically, but the best fitness is most important since the best individual is the solution of the problem.

4.2 Test functions and the experimental results

Many test functions are discussed in some papers [2, 7, 10, 11, 15]. In our experiments, eleven test functions are applied, and we show two of them. The space sizes and the numbers of variables of these test functions are listed on table 4.1. The parameter settings of these test functions are listed on table 4.2.

The five methods applied are as follows:

1. **SGA** (curve 1): The simple genetic algorithm.
2. **Unbiased** (curve 2): SGA + unbiased initialization.
3. **Uniform** (curve 3): SGA + uniform initialization.
4. **Extraction** (curve 4): SGA + extraction operator.
5. **Extr/Uni** (curve 5): SGA + extraction operator + uniform initialization.

In this subsection we analyze the average, maximum and best fitnesses and average and maximum curves of the experimental results.

1. ***Function f1***

This is a five-variable step function. The traditional search methods such as hill-climbing are hard to find the optimum since it is a discontinuous function. The goal is to find the minimum.

$$\sum_{i=1}^{5} \text{int}(x_i), \quad -5.12 \le x_i \le 5.12$$

Table 4.1 Space sizes of test functions

Function	*variables*	*Length(bit)*	*Space size*
f1	5	50	$1.13*10^{15}$
f2	2	44	$1.76*10^{13}$

Table 4.2 Parameter settings of test functions

Parameter	*f1*	*f2*
Run times	50	50
Population size	30	30
Individual length	50	44
Crossover rate	0.8	0.8
Mutation rate	.005	.005
Generation	100	300
Extraction reserve rate	0.6	0.6
Extraction replacement rate	0.6	0.6
Extraction comparison best#	5	5
Extraction interval	1	1

Table 4.3 Average, maximum and best fitness of f1

	Average	*Maximum*	*Best*
SGA	0.74314693	0.84101386	0.91320000
Unbiased	0.76146693	0.85939010	0.92480000
Uniform	0.79855063	0.89121188	0.94760000
Extraction	0.95272290	0.97743366	1.00000000
Extr/Uni	0.96695221	0.99019802	0.99960000

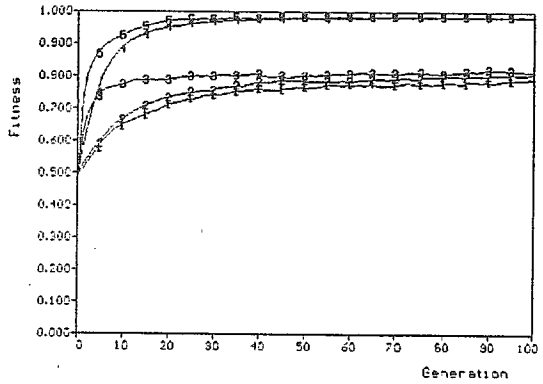

Figure 4.1 Average curve of f1

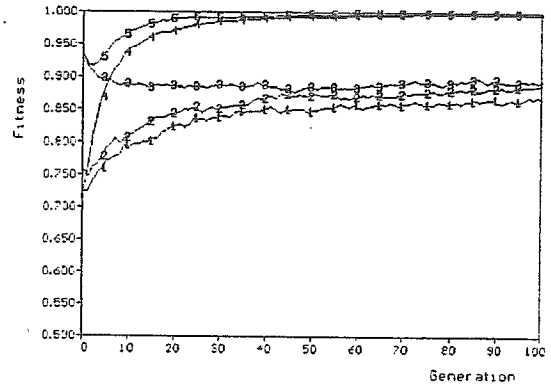

Figure 4.2 Maximum curve of f1

2. *Function f2*

This is a rapidly varying multimodal and extremely complex function. f2 has been employed in [6] and [16], where it is referred to as the "Sine envelope sine wave function". The following is the inverted version of the origin f2. When the global optimum is approached, the barrier height between adjacent maxima decreases [17]. The goal is to find the maximum.

$$0.5-\frac{\sin^2\sqrt{(x_1^2+x_2^2)}-0.5}{[1+0.001(x_1^2+x_2^2)]^2}\ ,\quad -100\le x_i\le 100$$

Table 4.4 Average, maximum and best fitnesses of f2

	Average	*Maximum*	*Best*
SGA	0.72371417	0.82247035	0.95497936
Unbiased	0.72882653	0.82834490	0.94970894
Uniform	0.74033659	0.84033818	0.96475124
Extraction	0.89428079	0.94170151	0.97079854
Extr/Uni	0.91402931	0.96271630	0.98176081

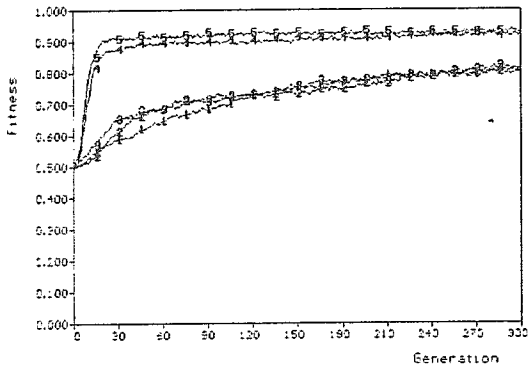

Figure 4.3 Average curve of f2

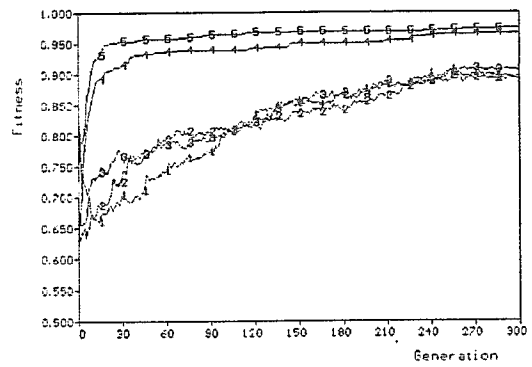

Figure 4.4 Maximum curve of f2

5. Conclusions

In this paper, we developed some novel methods including the extraction operator, the uniform initialization and the unbiased initialization. The **extraction** operator alters the inner structure of the individual, whereas the **uniform initialization** and the **unbiased initialization** methods modify the population initialization. All of these three methods are simple and can be combined with the simple genetic algorithms easily. Simulation results show that these methods exhibit an evident improvement of the GAs.

References

[1] T. Back and F. Hoffmeister, "Extended selection mechanisms in genetic algorithms," in *Proc. of the Fourth Intern. Conf. on Genetic Algorithms*, 1991.

[2] L. Booker, "Improving search in genetic algorithms," in *Genetic Algorithms and Simulated Annealing*, L. Davis, editor, Pitman Press, London, 1987.

[3] M.F. Bramlette, "Initialization, mutation and selection methods in genetic algorithms for function optimization," in *Proc. of the Fourth Intern. Conf. on Genetic Algorithms*, 1991.

[4] Y. Davidor, "Analogous crossover," in *Proc. of the Third Intern. Conf. on Genetic Algorithms*, 1989.

[5] L. Davis, "Adaptive operator probabilities in genetic algorithms," in *Proc. of the Third Intern. Conf. on Genetic Algorithms*, 1989.

[6] L. Davis, editor, "Handbook on Genetic Algorithms," Van Nostrand Reinhold, New York, 1991.

[7] K.A. DeJong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, 1975.

[8] L.J. Eshelman and J.D. Schaffer, "Preventing premature convergence in genetic algorithms by preventing incest," in *Proc. of the Fourth Intern. Conf. on Genetic Algorithms*, 1991.

[9] T.C. Forgarty, "Varying the probalility of mutation in genetic algorithms," in *Proc. of the Third Intern. Conf. on Genetic Algorithms*, 1989.

[10] D.E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, 1989.

[11] J.J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol.SMC-16, No.1, January/February 1986.

[12] J.H. Holland, "Adaptation in Natural and Artificial Systems," Ann Arbor: University Michigan Press, 1975.

[13] G.E. Liepins and M.R. Hilliard, "Greedy genetics," in *Proc. of the Second Intern. Conf. on Genetic Algorithms*, 1987.

[14] J.C. Potts, T.D. Giddens, and S.B. Yadav, "The development and evalutation of an improved genetic algorithm based on migration and artificial selection," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol.24, No.1, January 1994.

[15] J.D. Schaffer and A. Morishima, "An adaptive crossover distribution mechanism for genetic algorithms,"in *Proc. of the Second Intern. Conf. on Genetic Algorithms*, 1987.

[16] J.D. Schaffer, R.A. Caruana, L.J. Eshelman and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in *Proc. of the Third Intern. Conf. on Genetic Algorithms*, 1989.

[17] M. Srinivas and L.M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol.24, No.4, April 1994.

[18] K. Yamanoto, O. Incue, "New evolutionary direction operator for genetic algorithms (TN)," *AIAA Joural*, Vol.33, Iss.10, October 1995.