ISTANBUL UNIVERSITY ENGINEERING FACULTY
JOURNAL OF ELECTRICAL & ELECTRONICS

YEAR : 2002 (495-504)
VOLUME : 2
NUMBER : 2

UNIFORM POPULATION IN GENETIC ALGORITHMS

[1,2] Ali KARCI and Ahmet ARSLAN

[1,2] Firat Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliði Bölümü, 23119, Elazig/Turkey

[1]E-mail: akarci@firat.edu.tr [2]E-mail: aarslan@firat.edu.tr

ABSTRACT

The most of researchers dealing with genetic algorithms apply variation on the genetic operators. Some of them propose new genetic algotihm types. However, none of them deals with generating population of good quality, initially. In this paper, we propose a method for generating initial population and the method includes all types of chromosome encoding. The goodness of generated population by proposed method is also illustrated by applying this population and random initial population to multi-modal functions such as Griewank, Michalewicz and Rastrigin. For the sake of simplicity, all functions are selected as functions of two variables.

Keywords: *Genetic Algorithms, multi-modal functions*

1. INTRODUCTION

Genetic Algorithms (GAs) have been proven useful in solving NP-Complete and NP-Hard problems. GA is an evaluation method to generate random solutions for a problem and then applying genetic operators to random solutions to force them to be better solutions at each generation. So, sufficiently good solution/solutions can be obtained eventually.

In the conventional genetic algorithm model, a population of strings (chromosomes, individuals) codifying the possible solutions for the problem at hand passes through a cyclic process in which new candidates are constantly generated and evaluated according to some adequacy measure known as fitness. Parents are charged by computational operators, very much resembling natural evolutive reproduction, selection and mutation being progressively replaced by more adapted newcomers. The population fitness tends to converge on the course of the process and best or sub-best solutions are obtained at final stages.

The GA is a type of structured random search algorithm so-called by most of researchers who used GAs that mimics the process of biological evolution. The algorithm begins with a collection of parameter estimates (called a chromosome or individual) and each is evaluated for its fitness in

Received Date : 17.4.2002
Accepted Date: 29.5.2002

solving the given minimization or maximization task. At each generation (algorithm time-step), the most fit chromosomes are allowed to mate and bear offspring. The biological analogy suggests that such a procedure will be likely to lead to workable solution for complex non-linear problems.

A GA traditionally contains three types of operators: selection, crossover and mutation. A simple GA executes as follows:

a) Start with a randomly generated population of n kbit chromosomes. These are the candidate solutions to the problem.
b) Calculate the fitness F(x) of each chromosome x in the population.
c) Repeat the following steps until n offspring have been created.
    i. Select a pair of chromosome playing the role as parents. The probability of an individual been selected is usually a function of fitness. The fitter the individual is, the more likely it will be selected to reproduce.
    ii. With a probability P_c (the crossover rate), crossover the pair at a randomly chosen point to form two offspring. If no crossover takes place, form two offspring that are exact copies of their respective parents.
    iii. Mutate the some worse individuals in the population at each locus with probability P_m (the mutation rate) and place the resulting chromosomes in the new population.
d) Replace the current population with the new population.
e) Go to step b.

Random search algorithms have achieved increasing popularity as researchers recognise the shortcomings of calculus-based and enumerative schemes. Random walks and random schemes that search and save the best must be discounted because of efficiency requirements. Random searches, in the long run, can be expected to do no better than enumerative schemes. Random search methods are distinct from randomised techniques. A GA is an example of a search procedure that uses random choice as a tool to guide a highly exploitative search through a coding of a parameter space.

Many search techniques require auxiliary information in order to work properly. GAs have no need for all this auxiliary information; they are blind. They only require payoff values associated with individual strings in performing an effective search for better and better structures. This characteristic makes a GA a more canonical method than many search schemes.

Many researchers have tried to improve the GAs performance by handling some modifications on the genetic operators and analyzing chromosomes space properties: dealing with genotype-phenotype mapping [1, 2], analyzing schema theory at aim of catching some idea for improving GAs performance [3, 19]. Some researchers investigated the effects of GAs operators and tried to modify GAs operators [4-8, 15-18, 21]. Some people defined the new version of GAs [9-14, 20-22]. In noisy environment, fitness of an individual cannot be evaluated precisely, but its fitness has to be estimated [16].

Most of researchers used GAs applied modifications on the GA operators to improve the performance of GAs. However, there are some problems related to GAs such as to be trapped in local solution/solutions or diverging from best or sub-best solution. These are important points for improving the performance of GAs.

In this paper, we have proposed a new method for generating initial population [12,13]. This method uses divide-and-conquer to generate an initial population of good quality. By this method, chromosomes are distributed over chromosomes space, so, improving population for catching solution takes less time than random initial population. Because of chromosomes distributed over space, to be trapped in local solution is impossible. Consequently, best or sub-best solution can be obtained in this way.

This paper is organized as follows. Section 2 describes the method of generating homogeneous population and Section 3 illustrates the results of this method to some multi-modal functions and comparison of this method to random initial method. Finally, the paper is concluded.

2. UNIFORM POPULATION METHOD (UP)

UP is a method to generate a population of good quality in order to overcome some aforementioned problems of genetic search. This method works as follows. Initially, a dividing factor, let r denote dividing factor, is determined by user or program, then a chromosome is randomly generated and it is divided into r parts. There are 2^r ways to take complement of one part or more than one part. So, it is possible to get 2^r-1 chromosomes from one randomly generated chromosome. UP is described for values of r=1, 2, and 3 in the following parts.

When r=1:

A chromosome is generated randomly and if the dividing factor is 1, then the inversion of randomly generated chromosome will be another individual in the initial population. This case is shown in Fig. 1(a). In this case, if there are n randomly generated chromosomes, then the size of initial population will be 2n.

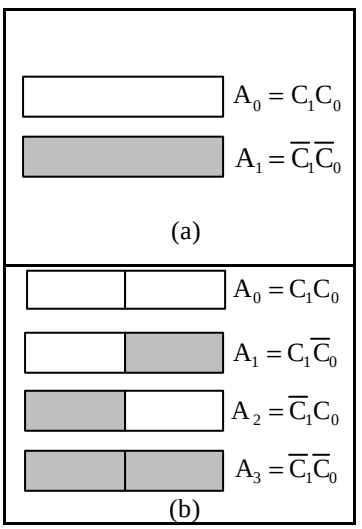


Fig. 1. Generating individuals from randomly generated individuals in case of r=1 and r=2.

When r=2:

If the dividing factor is 2, then, initially, a chromosome is randomly generated and then this chromosome is divided into two parts. The generated chromosome is an individual in population and its inversion is also another individual in initial population. After that, combining the inversion of rightmost part, and leftmost part without inversion will be an individual in initial population and combining inversion of leftmost part, and rightmost part without inversion will be another individual in population. So, there are three new generated individuals from randomly generated individual. If each randomly generated chromosome is divided into two parts and there are n randomly generated chromosomes, then there will be 4n individuals in the initial population, because there are three individual generated from each randomly generated chromosome. This case is shown in Fig. 1(b).

When r=3

If randomly generated chromosome is C_0 and it will be divided into three parts then deriving other individual from this chromosome is handled in same manner as starting from zero in binary coding and increasing by one in each step and this case is shown in Fig. 2. For each individual, derived individuals number is 7. So, if randomly generated chromosomes number is n, then the size of initial population will be 8n.

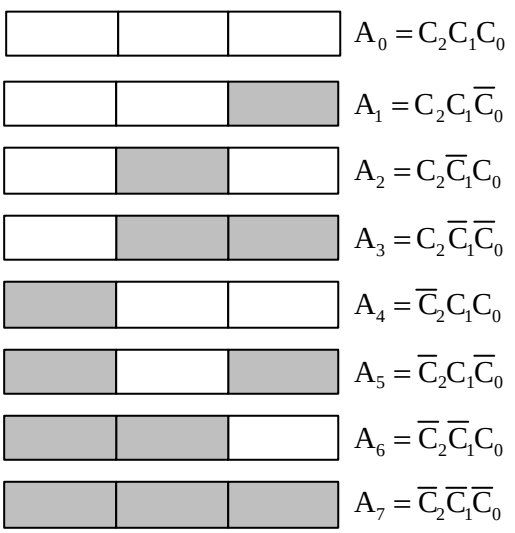


Fig. 2. Dividing initially random generated chromosome into three parts and deriving other individuals from this chromosome.

When r=General Case

If each randomly generated chromosome is divided into r parts, then the number of derived chromosomes from randomly generated chromosome is 2^r-1. So, the number of chromosomes in the initial population will be (the number of randomly generated chromosomes is n)

$$(2^r-1)n+n=n2^r \quad (1)$$

Hence, UP distributes the initial population individuals over hyperspace in uniform manner.
Up to this point, we mentioned about binary or integer encoding UP method. At this point we will briefly describe the other types of encoding types in UP.

Floating-Point Based Encoding. In this encoding method, each gene is represented by a floating-point number. Let R be a chromosome and $R=(r_0 r_1 r_2 ... r_{n-1})$ is a chromosome of length n and for $0 \le k \le n-1$. r_k is a floating point number, so, it consists of two parts: mantissa and exponent. The mantissa may be a fraction or an integer. The floating-point numbers used in this method are normalized. Let $r_k = r_{mn} r_{m(n-1)} ... r_{m1} . r_{es} r_{e(s-1)} ... r_{e1}$ and

$$R = \bar{r}_0 \bar{r}_1 ... \bar{r}_{n-1}$$

$$\bar{r}_k = (9 - r_{mn}) ... (9 - r_{m1}) . (9 - r_{es}) ... (9 - r_{e1}) \quad (2)$$

Each floating-point number in the chromosome is regarded as similar to bit in the binary encoding. Application of UP to integer-based encoding is same as floating-point type encoding. Another way for taking complement of a floating-point number is to subtract variable value from its upper bound.

String-Based Encoding. In this encoding method, each chromosome is represented by a sequence of alphabetic characters. Let R be a chromosome and $R=c_n c_{n-1} ... c_1$. So, the complement of R is

$$\bar{R} = \bar{c}_n \bar{c}_{n-1} \cdots \bar{c}_1 \text{ and for } 1 \le k \le n$$

$$ord(\bar{c}_k) = ord(Z) - [ord(c_k) - ord(A)] \quad (3)$$

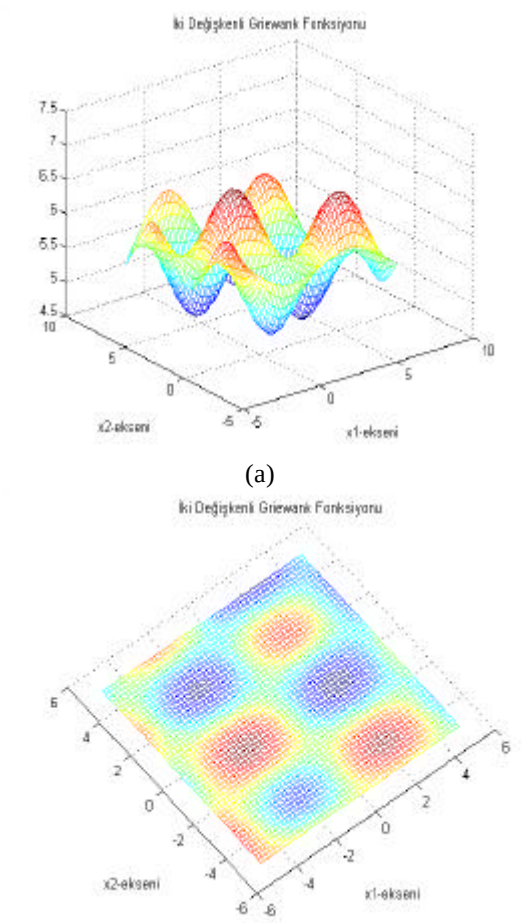
where function ord(.) retums the ASSCI value of parameter. Each character in the string is handled as similar to bit in the binary encoding.

So, UP method can be applied to each encoding types and the most important two problems of GAs can be dealt with. The key point is that dealing with GA problems will result in solution.

3. EXPERIMENTAL RESULTS

In order to illustrate the UP and random initial population method (RIP), we will use three multi-modal functions: Griewank, Michalewicz and Rastrigin Functions. Fig. 3, Fig. 4 and Fig. 5 denote Griewank, Michalewicz and Rastrigin Functions, respectively. The aim of selecting these functions is that these functions have more than one extreme point.

The Griewank function is



(a)

(b)

Fig. 3. Griewank Function.

$$f(x_1, \cdots, x_n) = \frac{1}{4000} \sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i - 100}{\sqrt{i}} \right) + 1 \quad (4)$$

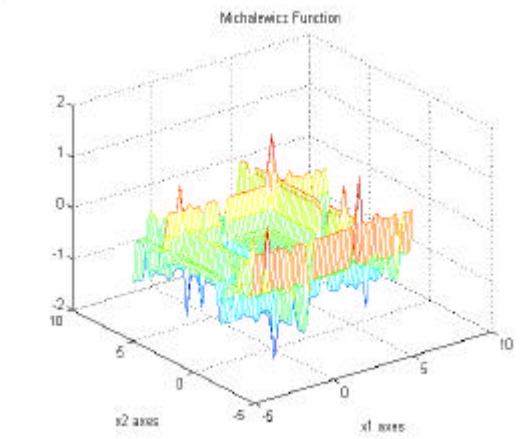
where n is the number of variables, and in this study, we used two-variable Griewank function and this function is depicted below.

$$f(x_1,x_2)=\frac{1}{4000}\left((x_1-100)^2+(x_2-100)^2\right)-\cos(x_1-100)\cos\left(\frac{x_2-100}{\sqrt{2}}\right)+1$$

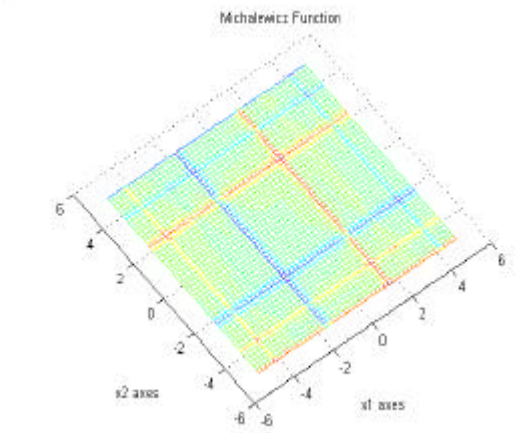
and $x_1,x_2\in[-5.0, 6.0]$.

The Michalewicz function is defined by equation (5)

$$f(x_1,\cdots,x_n)=\sum_{i=1}^{n}\sin(x_i)\sin^{20}\left(\frac{ix_i^2}{\pi}\right) \quad (5)$$



(a)



(b)

Fig. 4. Michalewicz Function.

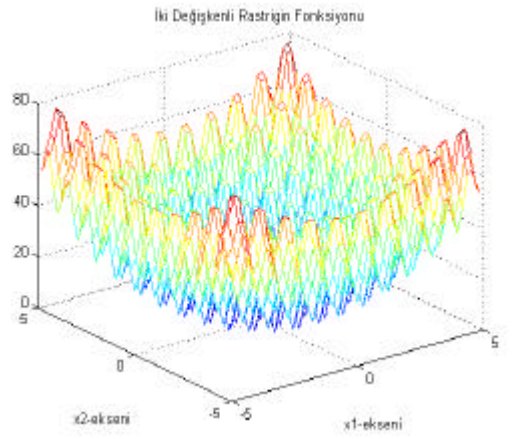
where n is the number of variables used in function and Michalewicz function used in this study is as follows.

$$f(x_1,x_2)=\sin(x_1)\sin^{20}\left(\frac{x_1^2}{\pi}\right)+\sin(x_2)\sin^{20}\left(\frac{2x_2^2}{\pi}\right)$$

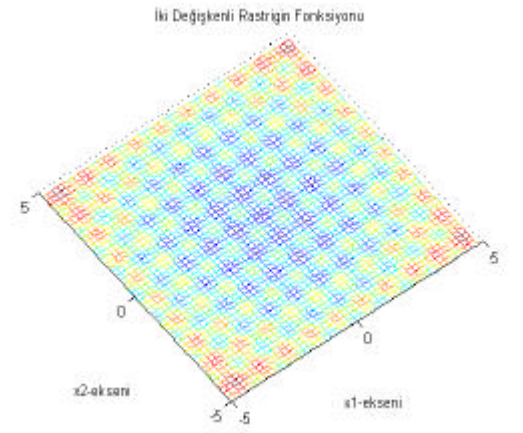
and $x_1,x_2\in[-3.0, 4.0]$, and the graph of this function is shown in Fig. 4.

The Rastrigin function is defined by equation (6).

$$f(x_1,\cdots,x_n)=10n+\sum_{i=1}^{n}\left(x_i^2-10\cos(2\pi x_i)\right) \quad (6)$$



(a)



(b)

Fig. 5. Rastrigin Function.

where n is the number of variables. The Rastrigin function used in this study is as follows.

$$f(x_1, x_2) = 20 + \left(x_1^2 - 10\cos(2\pi x_1)\right) + \left(x_2^2 - 10\cos(2\pi x_2)\right)$$

and x_1, $x_2 \in [-5.0, 6.0]$, and the graph of this function is shown in Fig. 5.

With respect to experimental results, UP and RIP can be compared in two ways: Based on the number of iterations and the obtained function results. We tried to obtained global minimum points of each function under the range of variables. Let C be a chromosome and then each chromosome represents variables of functions as follows.

$C=A_{2m}|A_{2e}|A_{1m}|A_{1e}$

where A_{2m} is mantissa of x_2, A_{2e} is exponentiation of x_2, A_{1m} is mantissa of x_1, A_{1e} is exponentiation of x_1, and | denotes concatenation of strings.

Fig. 6 shows the graphs of number of iterations and function values of both methods for Griewank function.

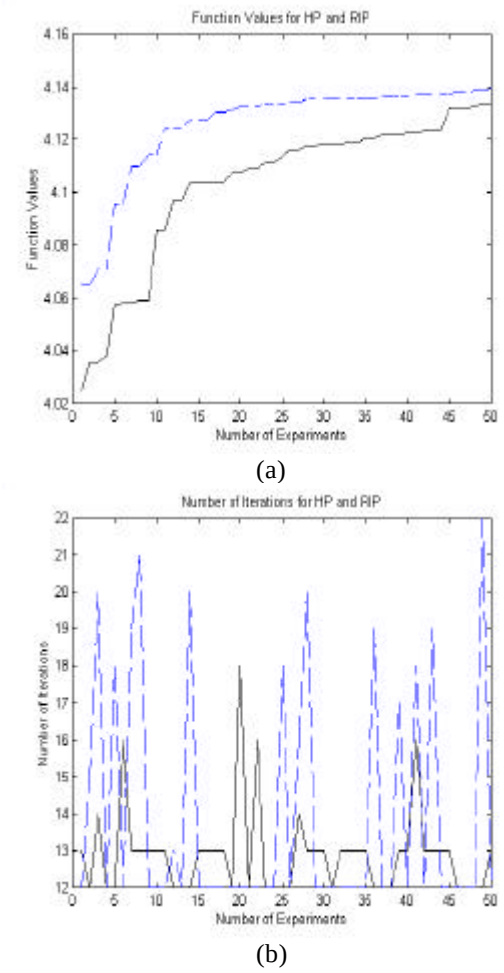


Fig. 6. (a) shows the values of function for both methods and (b) shows the number of iterations of both methods. Thick line (black) is beyond to UP and dashed line (green) is beyond to RIP. These figures were obtained after applying 50 times both methods to Griewank function.

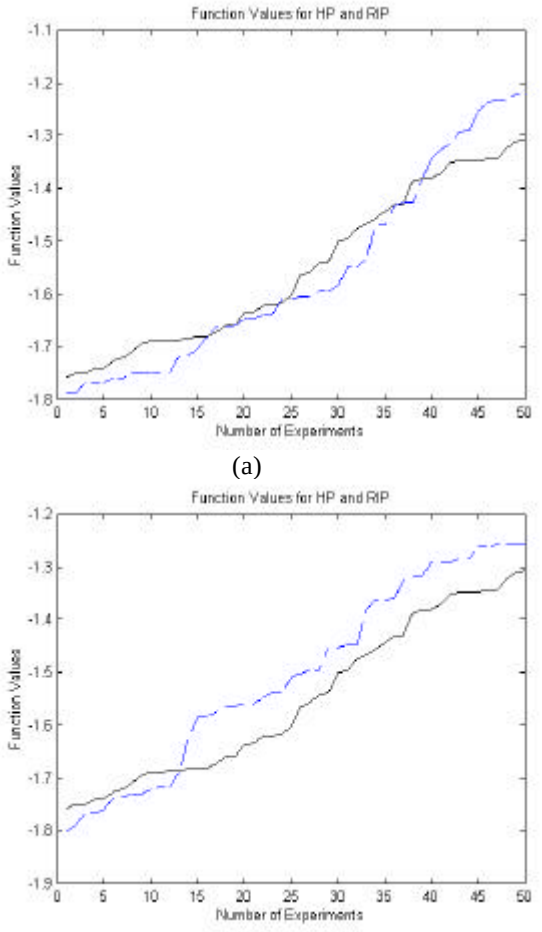


Fig. 7. (a) shows the function values of both methods and (b) shows the number of iterations of both methods. Thick line is beyond to UP and dashed line is beyond to RIP. These figures were obtained after applying 50 times both methods to Michalewicz function.

While applying both methods to Griewank function, selection type is radiused selection* crossover type is uniform, mutation type is random, the population size is 96, and r=5 for UP[1]. UP takes less number of iterations than RIP and it also found better solution than RIP. Without changing any genetic operators, both methods were applied to Michalewicz and Rastrigin functions. UP is again better than RIP with respect to optimal solution and number of iterations. The consequences are shown in Fig. 7 and Fig. 8, respectively. With respect to number of iterations, UP is 7.5846% better than RIP for Griewank function, UP is 6.6332% better than RIP for Michalewicz function, and UP is also 8.0721% better than RIP for Rastrigin function.

Theorem 1: Generating initial population by UP will result in a population of good quality.

Proof: In order to proving this theorem, let us investigate the expected value of initial population P. If r=2 then, we can generate 3 extra chromosomes from a randomly generated chromosome $X_1=A_1A_2$, and without losing generalization, let these chromosomes be $X_1=A_1A_2$, $X_2=A_1\overline{A}_2$, $X_3=\overline{A}_1A_2$, $X_4=\overline{A}_1\overline{A}_2$ and $|X_1|=n$. Then, the value of largest chromosome in this population is at most 2^n. So,

$$X_1+X_4=2^n \text{ and } X_2+X_3=2^n.$$

In the case of r=4, there are 15 extra chromosomes generated from randomly generated chromosome $X_1=A_1A_2A_3A_4$. Then

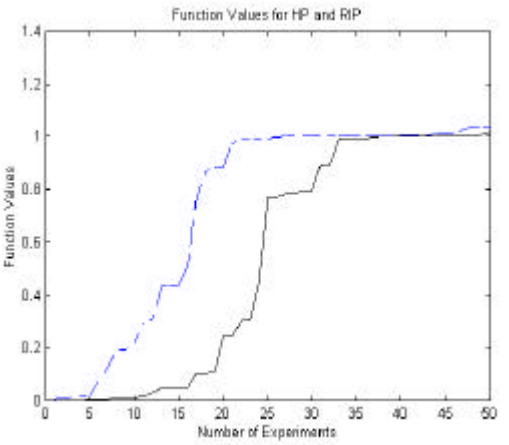
$$X_2=A_1A_2A_3\overline{A}_4 \qquad X_3=A_1A_2\overline{A}_3A_4$$
$$X_4=A_1A_2\overline{A}_3\overline{A}_4 \qquad X_5=A_1\overline{A}_2A_3A_4$$
$$X_6=A_1\overline{A}_2A_3\overline{A}_4 \qquad X_7=A_1\overline{A}_2\overline{A}_3A_4$$
$$X_8=A_1\overline{A}_2\overline{A}_3\overline{A}_4 \qquad X_9=\overline{A}_1A_2A_3A_4$$
$$X_{10}=\overline{A}_1A_2A_3\overline{A}_4 \qquad X_{11}=\overline{A}_1A_2\overline{A}_3A_4$$
$$X_{12}=\overline{A}_1A_2\overline{A}_3\overline{A}_4 \qquad X_{13}=\overline{A}_1\overline{A}_2A_3A_4$$
$$X_{14}=\overline{A}_1\overline{A}_2A_3\overline{A}_4 \qquad X_{15}=\overline{A}_1\overline{A}_2\overline{A}_3A_4$$
$$X_{16}=\overline{A}_1\overline{A}_2\overline{A}_3\overline{A}_4$$

and

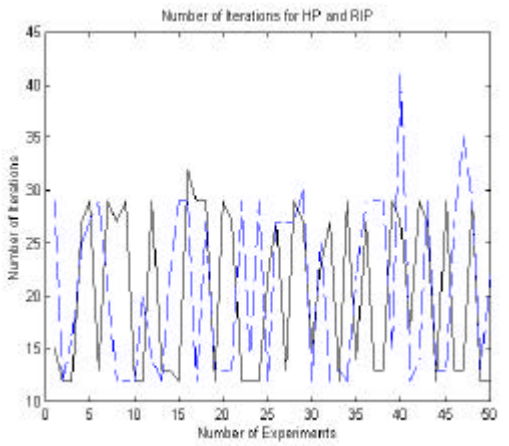
$X_1+X_{16}=2^n$ $X_2+X_{15}=2^n$ $X_3+X_{14}=2^n$
$X_4+X_{13}=2^n$ $X_5+X_{12}=2^n$ $X_6+X_{11}=2^n$
$X_7+X_{10}=2^n$ $X_8+X_9=2^n$

For all values of r, these equalities are held.

Hence, $E[P]=(2^n+2^n)/2=2^{n-1}$ where E[P] is the expectation of population.



(a)



(b)

Fig. 8. (a) shows the function values of both methods and (b) shows the number of iterations of both methods. Thick line is beyond to UP and dashed line is beyond to RIP. These figures were obtained after applying 50 times both methods to Rastrigin function.

The first point for illustrating goodness of population is its expectation and second point is to compute change coefficient and third moment of population. The variance of population is

$$\sigma^2(P)=\sum_{i=1}^{|P|}(P_i-E[P])^2$$

and then change coefficient σ(P) will be less than expectation of population. So,

$$\frac{\sigma(P)}{E[P]} < 1$$

This means that the population does not widen rapidly. Another important point is to denote the uniformity of population. If third moment of population is equal to zero, then the population is uniform. So,

$$\sigma^3(P) = \sum_{i=1}^{|P|} (P_i - E[P])^3$$

and it is trivial to observe that

$$P_1\text{-}E[P] = -(P_{|P|}\text{-}E[P])$$
$$P_2\text{-}E[P] = -(P_{|P|-1}\text{-}E[P])$$
$$\ldots\ldots\ldots\ldots$$
$$P_{|P|/2}\text{-}E[P] = -(P_{|P|/2+1}\text{-}E[P])$$

Hence, $\sigma^3(P)=0$.

So, the obtained population chromosomes are not crowded in one part of space and they are distributed over space uniformly/homogeneously ♦

Theorem 2: UP preserves at least one chromosome nearest to solution chromosome than RIP in general.

Proof: Let population P_1 be generated by UP and population P_2 be generated by RIP. $P_1 \cap P_2 = P_3$, and let S be solution chromosome. Then min $(H(P_1,S))$ and $\min(H(P_2,S))$ are minimum Hamming distances between P_1 and S, P_2 and S, respectively. If $|S|=n$, then maximum Hamming distance is n. P_1 preserves the chromosome of Hamming distance $n-\min(H(P_3,S))$ and this chromosome is not included in P_2 ♦

Important Point: Let variables be embedded into a chromosome as seen in Fig. 9. If each chromosome in UP is divided into r parts, and then there are r-1 dividing points in each chromosome. Dividing points separate two adjacent parts in a chromosome.

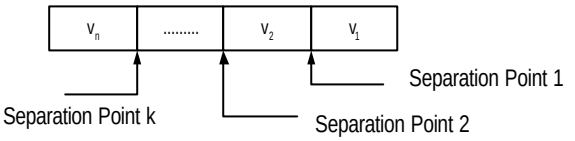


Fig. 9. Variables representation in each chromosome and separation points of variables coincide with dividing points in UP.

If dividing points coincide with separation points, then increasing r does not matter. In this case, population obtained by UP for each possible values of r is same the population obtained by UP for r=1. So, in fact, someone cannot get the performance of UP in this case. However, this case is rarely and it is not often met.

4. CONCLUSION

UP and RIP were applied to multi-modal functions. UP method distributes initial population over chromosomes space uniformly, so, time of GA progressing decreases and obtained function result is better than RIP result.

Initial population is distributed over chromosomes space and then solution point has at least one chromosome of δ-neighbourhood where δ=hamming distance between solution point chromosome and the nearest chromosome to solution chromosome. However, RIP method will not guarantee this case.

It can be seen from the expectation of population generated by UP, and its expectation is average of all encodable chromosomes in chromosomes space. Another important point is that, when all chromosomes in population have values 2^{n-1}, expectation of this population is also the average of all encodable chromosomes in chromosomes space. But the population generated by UP cannot have such a collection of chromosomes. This means that all chromosomes are distributed over chromosomes space.

REFERENCES

1. R.E. Keller, W. Banzhaf," Evolution of Genetic Code on a Hard Problem", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:50-56, July 7-11, 2001.
2. S. Luke, "When Short Runs Beat Long Runs", Proc. Of the Genetic and Evolutionary

Computation Conference, San Fransisco, California, pp:74-80, July 7-11, 2001.

3. R. Poli, N. F. McPhee, « Exact Schema Theory for GP and Variable-Length Gas w Homologous Crossover", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:104-111, July 7-11, 2001.
4. R. Poli, J.E. Rowe, N. F. McPhee, « Markov Chain Models for GP and Variable-Length Gas w Homologous Crossover", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:112-119, July 7-11, 2001.
5. W.M. Spears, "The Role of Mutation and Recombination in Evolutionary Algorithms", PhD dissertation at George Mason University, Fairfax, Virginia, 1998.
6. M. Srinivas, L.M. Patnaik, "Genetic Search : Analysis Using Fitness Moments", IEEE on Knowledge and Data Engineering, vol:8, no:1, pp:120-133, 1996.
7. P. J. Angeline, "Two Self-Adaptive Crossover Operations for Genetic Programming", Advances in Genetic Programming: Volume 2.
8. J.A. Vasconcelos, J.A. Ramirez, R.H.C. Takahashi, R.R. Saldanha, "Improvement in Genetic Algorithms", IEEE Trans. On Magnetics, vol:37, no:5, pp:3414-3417, 2001.
9. N.Y. Nikolaev, H. Iba, "Regularization Approach to Inductive Genetic Programming", IEEE Trans. On Evolutionary Computation, vol:5, no:4, pp:359-375, 2001.
10. B. Edmonds, "Meta-Genetic Programming: Co-evolving the Operators of Variation", Turk J. Elec. Engin. Vol:9, no:1, pp:13-29, 2001.
11. P.D. Stroud, "Kalman-Extended Genetic Algorithm for Search in Nonstationary Environments with Noisy Fitness Evaluations", IEEE Trans. On Evolutionary Computation, vol:5, no:1, pp:66-77, 2001.
12. A. Karcý, A. Çýnar, "Comparison of Uniform Distributed Initial Population Method and Random Initial Population Method in Genetic Search", The 15th International Symposium on Computer and Information Sciences, Istanbul, Turkey, 159-166, 2000.
13. A. Karcý, A. Arslan, "Bidirectional Evolutionary Heuristic for the Minimum Vertex-Cover Problem", Journal of Computers and Electrical Engineering, (will be appeared).
14. Y.-W. Leung, Y. Ewang, éAn Orthogonal Genetic Algorithm with Quantization for Global Numerical Optimization", IEEE Trans. On Evolutionary Computation, vol:5, no:1, pp:41-53, 2001.
15. H.E. Aguirre, K. Tanaka, S. Oshita, "Increasing the Robustness of Distributed Genetic Algorithms by Parallel Cooperative-Competitive Genetic Operators", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:195-202, July 7-11, 2001.
16. J. Branke,"Reducing the Sampling Variance When Searching for Robust Solutions", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:235-242, July 7-11, 2001.
17. M. Elhadef, D.A. Coley," Adaptive Mutation for Semi-Separable Problems", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:306-312, July 7-11, 2001.
18. J.E. Rowe, N.F. McPhee, "The Effect of Crossover and Mutation Operators on Variable Lengrh Linear Structures", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:535-542, July 7-11, 2001.
19. C. Schumacher, M.D. Vose, L.D. Whitley, "The No Free Lunch and Problem Description Length", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:565-570, July 7-11, 2001.
20. A. Simoes, E. Costa, "On Biologically Inspired Genetic Operators: Transformation in the Standard Genetic Algorithm", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:584-591, July 7-11, 2001.
21. J. Smith, "Modelling GAs with Self Adaptive Mutation Rates", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:599-606, July 7-11, 2001.
22. R. P. Srivastava, D.E. Goldberg, "Verification of the Theory of Genetic Algorithm Continuation", Proc. Of the Genetic and Evolutionary Computation Conference, San Fransisco, California, pp:623-630, July 7-11, 2001.

Ali Karcý received the B.S. degree in computer engineering and information science from Bilkent University in 1994 and the M.S. degree in computer engineering from Fýrat University in 1998. He is still a Ph.D. student of Electrical & Electronics Engineering at Fýrat University. His areas of interest include evolutionary computing, parallel computing, interconnection networks.

Ahmet Arslan received the B.S. degree in electrical and electronics engineering from Fýrat University in 1984, the M.S. degree from Fýrat University in 1987 and Ph.D. degree in computer engineering and information science from Bilkent University in 1992. His areas of interest include genetic optimization, computer graphics, neural networks, agent-based computing and machine learning.