THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2003 ISSN 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS

A NEW METHOD OF GENE CODING FOR A GENETIC ALGORITHM DESIGNED FOR PARAMETRIC OPTIMIZATION

Radu Belea* **and Liviu Beldiman***

Department of Control System and Industrial Informatics, University "Dunărea de Jos" of Galati, Faculty of Electrical Engineering and Computer Science, Domneasca Street 47, 6200, Galați, Romania. Phone: (+40) 236-414872, Phone+Fax: (+40)236-460182, E-Mail: Radu.Belea@ugal.ro., Liviu.Beldiman@ugal.ro.

Abstract: In a parametric optimization problem the genes code the real parameters of the fitness function. There are two coding techniques known under the names of: *binary coded genes* and *real coded genes*. The comparison between these two is a controversial subject since the first papers about parametric optimization have appeared. An objective analysis regarding the advantages and disadvantages of the two coding techniques is difficult to be done while different format information is compared. The present paper suggests a gene coding technique that uses the same format for both binary coded genes and for the real coded genes. After unifying the real parameters representation, the next criterion is going to be applied: the differences between the two techniques are statistically measured by the effect of the genetic operators over some random generated fellows.

Keywords: binary coded genes, exploration, exploitation, random initialized genes, uniform crossover, HUX crossover, arithmetic crossover.

1. INTRODUCTION

Everything that can be found in a numerical computer, numerical information, graphical information, texts, programs, operating systems, algorithms etc. is coded with binary digits. That is why the expressions *binary coded genes* and *real coded genes*, even if they are very often used, are pleonasms. It is proper to speak about *fixed-point real numbers* and *floating-point real numbers*, but these expressions aren't consecrated in the genetic algorithms' literature.

In literature is often used a word-game: *exploration-exploitation*. The verb *to explore* means to search over large areas, while the substantive *exploit* also has the meaning of mining exploitation or mine. In this word-game the verb *to exploit* means to search with small steps. A genetic algorithm has good results if it combines large areas searching with small steps searching.

In Crawford [1977] there are enounced eight guidelines that guide a new crossover operator design. Two of them are important for the present paper.

- **Guideline 5:** the crossover operator should explore, not exploit.
- **Guideline 8:** In general, small (large) changes in genotype should produce small (large) changes in phenotype.

In a genetic algorithm, the exploration is good if the population covers, in time, uniformly the search space, and if the number of visited points is as big as possible. According to guideline 5 (Crawford, [1977]), the exploration success depends firstly on

This paper was recommended for publication by Rustem POPA

the crossover operator used, and then on the space shape and the selection method used.

The paper is organized as it follows: in section 2 there is briefly presented the arithmetic crossover. In section 3 there are presented the binary representation techniques of real numbers. In section 4 it is explained a new gene coding method, method that unifies binary coded genes with real coded genes. In section 5 it is used the bits histogram to test if the genetic operators were programmed correct. In section 7 it is used the histogram of Hamming distance between parents and children in order to appreciate the crossover operators quality. A summary of the results will be done in the last section of the paper.

2. CROSSOVER BETWEEN REAL NUMBERS

In real coded genes genetic algorithms, the parents chromosomes $p_1[k]$, $p_2[k]$ and the children chromosomes $o_1[k]$, $o_2[k]$, $k=1,\ldots,K$ are gene arrays. The most popular crossover variant between real numbers is the arithmetic crossover. Genes situated in the k position of the children chromosomes are calculated as it follows:

$$(1)\quad \begin{aligned} o_1[k] &= \alpha \cdot p_1[k] + (1-\alpha)\cdot p_2[k] \\ o_2[k] &= (1-\alpha)\cdot p_1[k] + \alpha \cdot p_2[k] \end{aligned},$$

where $\alpha \in [0,1)$ is a random real number uniformly distributed. The arithmetic crossover has a major disadvantage: it doesn't explore the whole space.

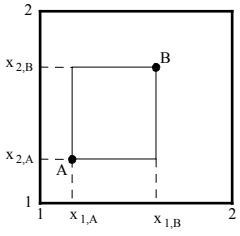

Fig.1. The effective space explored by the arithmetic crossover

In figure 1 it is presented a two-dimensional searching space $(x_1, x_2) \in [1,2)\times[1,2)$. The arithmetic crossover produces two children situated in the rectangle delimited by the $(x_{1,A}, x_{2,A})$ and $(x_{1,B}, x_{2,B})$ points that corresponds to the parents chromosomes and it doesn't explore the whole space. In order to avoid this fact, the intermediate crossover may be used:

$$(2)\quad \begin{aligned} o_1[k] &= p_1[k] + \alpha_1 \left(p_2[k] - p_1[k]\right) \\ o_2[k] &= p_2[k] + \alpha_2 \left(p_2[k] - p_1[k]\right) \end{aligned}$$

where $\alpha_1, \alpha_2 \in [-\delta, 1+\delta)$ are two random numbers and $\delta = 0.25$ is a number chosen by the programmer. In the case $\delta = 0.25$ it is necessary to verify for each gene if the result obtained after the crossover didn't overflow the searching space.

3. THE BINARY REPRESENTATION OF REASL NUMBERS

In order to record the exact value of a real number an infinite number of digits is needed. For example, there are known some thousands decimals of the number π but its exact value isn't known. That means that the value of a real number stored in a numeric calculator is truncated, as it is represented with a finite sequence of bits.

A finite sequence of bits $b_p b_{p-1}\ldots b_1 b_0 b_{-1}\ldots b_{q+1} b_q$ is a fixed-point unsigned real number. The value of this number is:

$$(3)\quad V = \sum_{i=-q}^{p} 2^i b_i \ .$$

Consequently, with $p+q+1$ binary digits real numbers can be represented in the domain $0 \le V \le 2^p$ with a resolution of 2^{-q}. The resolution is the difference between two consecutive real numbers in the respective representation, and the maxim representation error is half of the resolution, that is 2^{-q-1}.

A floating-point real number can be represented as it follows:

$$(4)\quad V = (-1)^S \cdot 2^E \cdot M \ ,$$

where $S \in \{0,1\}$ is the sign bit, E is a signed integer called exponent, and M is fixed point real number called mantissa. At the end of a floating point operation, the numbers E and M are arranged so that the mantissa has only one binary digit equal to 1 in front of the point. This operation is called the mantissa normalization. In the case of working with a normalized mantissa, it isn't necessary to store the digit in front of the point. The value of the real number is:

$$(5)\quad V = (-1)^S \cdot 2^{N-n_{off}} \cdot (b_{msb}.F) \ ,$$

where F is the fractional part of the mantissa. The bit b_{msb} is the most significant bit of the mantissa. In the case of floating point real numbers with normalized mantissa $b_{msb} = 1$. The exponent $E = N - n_{off}$ is stored with a gap of n_{off} so that the number N never has all the bits null. The number

n_{off} is chosen so that $-n_{off} \le E \le n_{off}$. It can be observed that the real number 0 cannot be represented in a floating-point real number with normalized mantissa. From this reason, the number real 0 is coded by storing all the bits null.

	K	S	N	b_{msb}	F	n_{off}
Single	4	b_1	$b_2 \cdots b_9$	-	$b_{10} \cdots b_{32}$	127
Real	6	b_1	$b_{40} \cdots b_{..}$	-	$b_2 \cdots b_{39}$	125
Double	8	b_1	$b_2 \cdots b_{11}$	-	$b_{12} \cdots b_{64}$	1023
Comp	10	b_1	$b_2 \cdots b_{15}$	b_{16}	$b_{17} \cdots b_{80}$	6535

Table 1

The bits signification of the four representations of the real numbers in the IBM-PC computers is given in Table 1. The real number is stored as a binary digits array of form: $b_1 b_2 b_3 b_4 \cdots b_{8K}$ where K is the byte number of the real number representation, S is the sign bit, N is a bits array that stores the exponent, b_{msb} is the most significant bit of the mantissa, F is a bits array that stores the fractional part of the mantissa of the real number, and n_{off} is the gap used to compute the exponent.

4. A NEW METHOD OF GENE CODING

The major disadvantage of the floating-point representation is the dependency between the number value and the bits weight. That's why the mutation and the crossover operators cannot be used the way they were defined in the case of binary coded genes. Though, in the representation of floating-point real numbers with normalized mantissa there are intervals $x \in [2^k, 2^{k+1})$, $k \in Z$, where all the real numbers from this interval have the same exponent and the mantissa's bits with the same position have the same weight. For example, the real parameter $x \in [x_{\min}, x_{\max})$ can be normalized with the function:

$$(6)\ [x_{\max}, x_{\min}) \to [1,2)\ ,\quad \tilde{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} + 1$$

In the next examples the "single" type is used, stored on 32 bits. The numbers in the interval [1,2) have $S = 1$, $N = 01111111$, so the exponent is $E = N - 127 = 0$, and the mantissa takes values starting with the combination $1.000\ldots0$ to the combination $1.111\ldots1$. The numbers are normalized, so there are stored only the first 23 bits from the fractional part of the mantissa. This representation has the following advantages:

- the chromosome is an array of real numbers;
- there can be used genetic operators defined for binary coded genes, but also genetic operators defined for real coded genes;
- there is no need of a function for decoding the genetic information. It is sufficient to denormalize the gene.
- All the arithmetic operations are realized by the co-processor, fact that speeds up the genetic algorithm speed.

This representation imposes some restrictions when programming genetic operators for binary coded genes. The mutation and the crossover must not modify the bits that carry the sign and the exponent. For all the numbers normalized to the interval $\tilde{x} \in [1,2)$ on four bytes, the bits that shouldn't be modified are $b_1 \ldots b_9 = 001111111$.

5. TESTING GENETIC OPERATORS WITH MONTE CARLO METHODS

In order to verify the correctitude of the genetic operators implementation, it is necessary to make some statistic tests for the random initialization, the mutation and the crossover. The gene random initialization was implemented with the instruction:

$$(7)\ g := Random + 1;$$

where the function *Random* generates a random number in the interval $[0,1)$. In Figure 2 there is presented a bit histogram for 1000 genes random initialized.

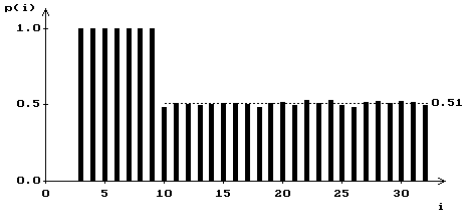

Fig. 2. Statistic test for random initialization

In the figure, on the abscissa the bits are placed in the order from table 1, and on the ordinate is the probability of the bits that have the value 1. The random initialization is correct if $b_1 = b_2 = 0$, $b_2 \ldots b_9 = 1$, and any bit from $b_{10} \ldots b_{32}$ takes the value 1 with a probability of 0.5. The mean value was calculated only for the bits $b_{10} \ldots b_{32}$ that can modify their content.

In Goldberg [1991] the next recommendation is enounced: "the mutation probability must not be affected by the bits number that code the gene". In order to fulfill this recommendation and to avoid excessive use of the random generator first it is calculated n, the number of the bits that will suffer a mutation with the instruction:

$$(8)\quad n := Trunc(23 * pm + Random);$$

where *Trunc* is the round function by truncation to the lower value, 23 is the number of bits of the fractional part of the mantissa, *pm* is the mutation probability, and *Random* is the function that generates random numbers uniformly distributed in the interval [0,1). Then the random numbers generator is used n times more to establish the bits that are going to be modified.

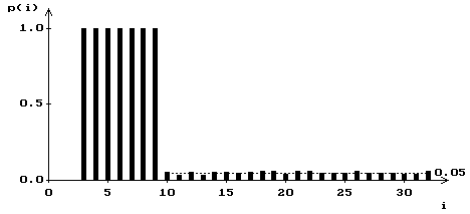

Fig. 3. Statistic test for mutation

In figure 3 is presented a histogram of the bits that code the gene for 1000 applications of the mutation operators with $p_m = 0.05$. The gene was initialized with the value $g = 1.0$. The mutation operator is programmed correctly if it results from the histogram that every bit from $b_{10} \ldots b_{32}$ takes the value 1 with the probability p_m.

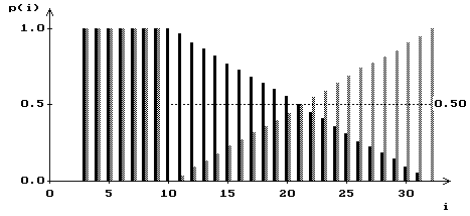

Fig. 4. Statistic test for crossover

In figure 4 it is presented a histogram of the bits that code the g_1 genes (gray bars) and the g_2 genes (black bars). The histograms were realized after the crossover was applied 1000 times over the genes: $g_1 = 1.99\cdots$ and $g_2 = 1.0$. It was studied the one-point crossover as it was described in Goldberg [1991]. The crossover operator is programmed correctly if the bits $b_{10} \ldots b_{32}$ of the g_1 histogram take the value 1 with an increasing probability from 0 to 1, and the bits $b_{10} \ldots b_{32}$ of the g_2 histogram take the value 1 with a decreasing probability from 1 to 0.

6. THE CROSSOVER OPERATOR QUALITY MEASURING METHOD

The Hamming distance between two chromosomes was used to analyze the uniform crossover even from the first papers of Booker or Eshelman where there were suggested the uniform crossover and the HUX (highly disruptive form of crossover) crossover. Let it be two genes $g_1 = 00111$ and $g_2 = 10110$. The hamming distance between g_1 and g_2 is:

$$(9)\quad d_h = bc(p_1 \text{ xor } p_2) = bc(00111 \text{ xor } 10110) = bc(10001)$$

where the logical operator "*xor*" is applied to the bits with the same position, and $bc(\cdot)$ is a function that counts the non-zero bits in a string of bits. A first result obtained from using the Hamming distance in the analyze of the crossover operator is the Booker observation:

$$(10)\quad \begin{array}{ll} d_h(p_1, p_2) = 0 & \Rightarrow \begin{cases} o_1 = p_1 \\ o_2 = p_2 \end{cases} \\ d_h(p_1, p_2) = 1 & \Rightarrow \begin{cases} o_1 = p_2 \\ o_2 = p_1 \end{cases} \end{array}$$

where o_1 and o_2 are called *substitute offspring*.

Correlating the guidelines 5 and 8 presented by Crawford with Booker observation, the next criteria results:

Criteria: the crossover operator explores the better the parameters space if the Hamming distances between the parents and the offspring chromosomes are statistically the larger.

Let it be a species of solutions that have a gene coded with the method suggested in section 4 of the present paper. It is noted with p_1, p_2 the parents chromosomes and o_1, o_2 the offspring chromosomes. Rana [1999] notices that:

$$(11)\quad \begin{array}{l} d_h(p_1, o_1) = d_h(p_2, o_2) \\ d_h(p_1, o_2) = d_h(p_2, o_1) \end{array}.$$

In Rana [1999] it is covered the set of the independent chromosomes and the histogram of the parents-offspring Hamming distances is realized. The Hamming distance between parents and offspring was calculated with the formula:

$$(12)\quad d_h(p, o) = \min\left(d_h(p_1, o_1) + d_h(p_1, o_2)\right)$$

The quality of the crossover operator is appreciated considering the probability of substitute offspring

apparition and the mean value of the distances calculated with the formula (12).

In the present paper there are considered a set of random initialized chromosomes and the Hamming distance between parents and offspring is calculated with the formula:

$$(13)\ d_h(p,o) = \frac{d_h(p_1,o_1) + d_h(p_1,o_2)}{2}.$$

The quality of the crossover operator is appreciated considering the mean value and the dispersion of the distances calculated with formula (13).

Moreover, this method is extended to the comparison of the binary crossover with the arithmetic crossover and to the comparison of the exploration realized by the arithmetic crossover with the exploration realized by the random initialization of the population.

The analyze method consist in random initialization of the parents genes and then two different crossover operators are applied and two pairs of children are obtained. Then, for each crossover operator, it is calculated the Hamming distance between parents and offspring. The last operations are repeated for a large number of pairs of genes random initialized, and then the Hamming distance histograms are graphically represented.

7. EXPERIMENTAL RESULTS

One point crossover is largely presented in Goldberg [1991]. In the case of uniform crossover, the probabilities of every bit transfer from one parent to offspring and the bit transfer in the same position, from the other parent is 0.5.

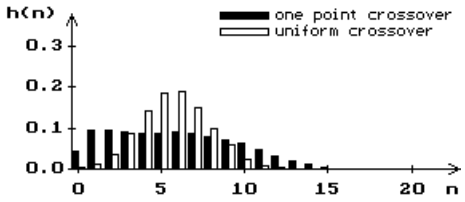

Fig. 5. The comparison between one point crossover and uniform crossover

In figure 5 there are comparatively presented the Hamming distance histograms between parents and offspring for one point crossover and uniform crossover. Summing the two samples from the figure 5 13.3% from the offspring obtained through one point crossover are substitute offspring, while only 1.23% of substitute offspring obtained after the uniform crossover, so the quality of the exploration realized by the uniform crossover is better because the probability of substitute offspring appearance is smaller.

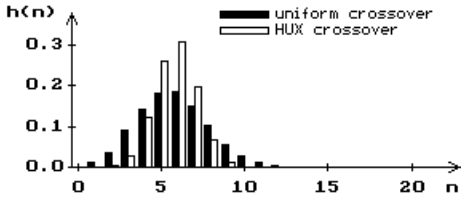

Fig. 6. The comparison between uniform crossover and the HUX crossover

In figure 6 there are comparatively presented the Hamming distance histograms between parents and offspring for uniform crossover and the HUX crossover. The HUX crossover is a uniform crossover variant that reduces the distance dispersion between the parents p_1, p_2 and the offspring o_1, o_2 imposing:

$$(14)\ \begin{matrix} d_h(p_1,o_1) = d_h(p_1,p_2)\ \text{div}\ 2 \\ d_h(p_2,o_2) = d_h(p_1,p_2) - d_h(p_1,o_1) \end{matrix},$$

the "*div*" operator from (12) representing the integer division without rest. The mask of the moved bits m_c, of the HUX crossover is obtained from the Hamming mask $m_h = p_1$ xor p_2, from which half of the bits with the value 1 are removed. Of course, the positions of these bits are randomly chosen. For 23 bits genes, the probability of substitute offspring decrease from 1.23% in the case of uniform crossover to 0.01% in the case of HUX crossover.

If the method suggested in section of the present paper is used, then the parents chromosomes $p_1[k]$, $p_2[k]$ and the offspring chromosomes $o_1[k]$, $o_2[k]$, $k = 1,\ldots,K$ are gene arrays and if it is performed the arithmetic crossover with the formula (1), where $\alpha \in [0,1)$ is a random real number, it results that:

$$(15)\ \begin{cases} p_1[k] \in [1,2) \\ p_2[k] \in [1,2) \end{cases} \Rightarrow \begin{cases} o_1[k] \in [1,2) \\ o_2[k] \in [1,2) \end{cases},$$

that means that the bits of a gene doesn't modify their weight in the gene value (see section 4). So, the arithmetic crossover can be compared to any of the binary crossover variants.

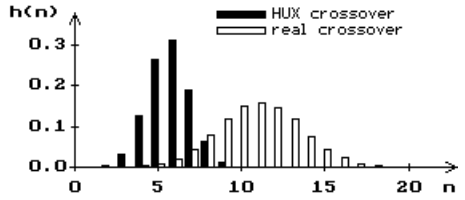

Fig. 7. The comparison between the HUX crossover and the arithmetic crossover

In figure 7 there are comparatively presented the Hamming distance histograms between parents and offspring for the HUX crossover and the arithmetic crossover. It can be noticed that the Hamming distances between parents and offspring are bigger in the case of the arithmetic crossover.

In the introductory part it was stated that the exploration is better if the number of points "visited" by the fellows is bigger. This "visitation" of the searching space starts in the moment of the random initialization of the genetic algorithm. The coding method suggested in section 4, together with the analyze method formerly used allow the comparison of the performances of the exploration realized by the random initialization with the exploration realized by the crossover genetic operator.

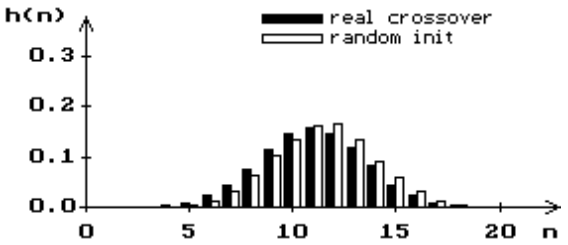

Fig. 8. The comparison between the exploration realized by the arithmetic crossover and the exploration realized by the random initialization of the genes

In figure 8 there are presented two histograms: the histogram of the Hamming distances between parents and offspring in the case of arithmetic crossover and the histogram of the Hamming distances between two randomly initialized genes. It can be notices that the arithmetic crossover is almost as efficient as the random initialization of the genes.

8. CONCLUSIONS

From the figures 5, 6 and 7 it results that due to the criteria enounced in section 6, the one-point crossover has the weakest performance and the performances increase from the uniform crossover, the HUX crossover to the real numbers crossover.

The arithmetic crossover (see formula (1)) realize a very good search when the population is concentrated in an ecological niche, improving the exploring process (see figure 7, section 7), but it doesn't explore the whole searching space (see figue1, section 2). So, the arithmetic crossover doesn't fulfill the Crawford's guideline 5.

The unwanted result from the figure 2 suggests a combined using of the uniform crossover or the HUX crossover with the arithmetic crossover. In this way, the arithmetic crossover reinitializes the bits of a part of the population, preserving the good solutions near the optimum point, while the binary crossover explores very well the searching space.

8. REFERENCES

Bäck T. [1997]. *Principles of Evolutionary Computation*. EUFIT 97, September 8-11, 1997, Aachen, Germany p. 759-763.

Crawford K. D. [1997*]. How one go developing a new crossover operator with an a priori expectation of its merit* ? Report at university of Tilsa 1997.

Eshelman L. J. [1991]. *The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. "Foundations of genetic Algorithms"*. Edited by Gregory Rowlins Morgan Kaufmann Publishiers, Inc. 1991.

Goldberg D. E. [1990]. *Real-coded Genetic Algorithms, Virtual Alphabets*. University of Illinois at Urbana-Champaign, Technical Report no. 90001, September 1990.

Goldberg D. E. [1991]. *Genetic Algorithms*. Addison-Wesley Usa, (15767), 1991. Traducere în limba franceză, Copyright © Juin 1994 Editions Addison-Wesley France, S. A.

Michalewicz Z. [1994]. *Genetic Algorithms + Data Structures = Evolution Programs*. © Springer-Verlag Berlin Heidelberg 1994.

Rana S. [1999]. *Disertation. Examining The Role of Optima and Schema Processing in Genetic Search*. For the Degree of Doctor of Philosophy, Colorado State University, Fort Colins, 1999.

Wright A. H. [1991]. *Genetic Algorithms for Real Parameter Optimisation*. "Foundations of genetic Algorithms". Edited by Gregory Rowlins Morgan Kaufmann Publishers, Inc. 1991.

***[1985]. ANSI/IEEE 754-1985 Standard.