New codification for genetic algorithms with crossed binary coding

Youness EL Hamzaoui, José A. Hernandez

Centro de Investigación en Ingeniería y Ciencias Aplicadas (CIICAp), Universidad Autónoma del Estado de Morelos (UAEM); Av. Universidad No.1001 Col. Chamilpa, C. P. 62209, Cuernavaca, Morelos, México.
{youness, alfredo}@uaem.mx

(Paper received on September 09, 2010, accepted on October 20, 2010)

Abstract. Genetics Algorithms (GAs) are based on the principles of Darwins evolution which are applied to the minimization complex function successfully. Codification is a very important issue when GAs are designed to dealing with a combinatorial problem. An effective crossed binary method is developed. The GAs have the advantages of no special demand for initial values of decision variables, lower computer storage, and less CPU time for computation. Better results are obtained in comparison the results of traditional Genetic Algorithms. The effectiveness of GAs with crossed binary coding in minimizing the complex function is demonstrated.

Keywords: Genetic Algorithms, crossed binary coding, mathematical functions.

1 Introduction

In mathematics, a function is a relation between a given set of elements called the domain. and a set of elements called the codomain. The function associates each element in the domain with exactly one element in the codomain. The elements so related can be any kind of thing (words, objects, qualities) but are typically mathematical quantities, such as real numbers. The concept of function can be extended to an object that takes a combination of two (or more) argument values to a single result. This intuitive concept is formalized by a function whose domain is the Cartesian product of two or more sets. A linear function, whose graph is a straight line which has an equation that can be written in the form of $y = mx + b$. However, equations whose graphs are not straight line or they are called nonlinear functions, some non linear functions have specific names. For example, a quadratic function is non linear and it has an equation in the form of : $y = ax^3 + bx + c$, where $a \neq 0$. Another nonlinear function is a cubic function which has an equation in the form of: $y = ax^3 + bx^2 + cx + d$, where $a \neq 0$. In mathematics and computer science, optimization, or mathematical programming, refers to choosing the best element from some set of available alternatives. Combinatorial optimization is a branch of optimization. Its domain is optimization problems where the set of feasible solutions is discrete or can be reduced to a discrete one, and the goal is to find the best possible solution. In the simplest case, this means solving problems in which one seeks to minimize or maximize a real function by systematically choosing the values of real or integer variables from within an allowed set. This formulation, using a scalar, real-valued objective function which is probably the simplest example; the generalization of optimization theory and techniques to other formulations comprises a large area of applied mathematics. More generally, it means finding "best available" values of some objective function given a defined domain, including a variety of different types of objective functions and different types of domains.

The increasing importance of nonlinear programming with mathematical programming (MP) is commonly used [1]. Because of the nonlinearity nature in minimization the complex function, unbearable long computation time will be induced by the use of MP when the nonlinear function is somewhat complicated [2]. Severe initial values for the optimization variables are also necessary. Moreover, with the increasing size of the nonlinear function, MP will be futile. Heuristics needs less computational time, and severe initial values for optimization variables are not necessary, but it may end up with a local optimum due to its greedy nature. Also, it is not a general method with respect to the fact that special heuristic rules will be needed for a special problem [3]. Many authors applied traditional Genetics Algorithms (GAs) to solve the problem. Traditional GAs performs effectively and gives a solution within 0.5% of the global optimum. However, traditional GAs has the disadvantage of long searching time and so needs more CPU time. The application of GAs analysis to solve complicated problems has been the subject of numerous review articles. The article published by Çelebi Mehmed describes the new approach based on two explicit rules of mendel experiments and mendel's population genetics for the genetic algorithm [4]. However, we have seen, to solve the proposed problem more effectively, we apply GAs with crossed binary coding method which is developed, an intelligent problem-solving method that has demonstrated its effectiveness in solving complicated problem, and satisfactory results are obtained.

The rest of this paper is organized as follows. Section 2 presents the methodology including the new codification to demonstrate the effectiveness of GAs with crossed binary coding in solving the proposed problem, section 3 presents four problems with three objective functions and their computations results using GAs with crossed binary coding in comparison with traditional GAs are also given. Finally, the conclusions on this work are drawn.

2 Methodology

2.1 Genetics Algorithms

The term genetics algorithms, almost universally abbreviated now a days to GAs, was first used by John Holland and his colleagues [5]. A genetics algorithms is a search technique used in computing to find exact or approximate solutions to optimization and search problems, however the canonical steps of the GAs can be described as follows: The problem to be addressed is defined and captured in an objective function that indicated the fitness of any potential solution.

A population of candidate solutions is initialized subject to certain constraints. Typically, each trial solution is coded as a vector ***X***, termed a chromosome, with elements being described as solutions represented by binary strings. The desired degree of precision would indicate the appropriate length of the binary coding. Each chromosome, Xi, $i = 1, \ldots, P$, in the population is decoded into a form an appropriate for evaluation and it is then assigned a fitness score, $\mu(Xi)$ according to the objective.

Selection in genetics algorithms is often accomplished via differential reproduction according to fitness. In a typical approach, each chromosome is assigned a probability of reproduction, P_i , $i = 1, \ldots, P$, so that its likelihood of being selected is proportional to its fitness relative to the other chromosomes in the population. If the fitness of each chromosome is a strictly positive number to be maximized, this is often accomplished using roulette wheel selection [6]. Successive trials are conducted in which a chromosome is selected, until all available positions are filled. Those chromosomes with above-average fitness will tend to generate more copies than those with below-average fitness.

According to the assigned probabilities of reproduction, P_i , $i = 1, \ldots, P$, a new population of chromosomes is generated by probabilistically selecting strings from the current population. The selected chromosomes generate "offspring" via the use of specific genetic operators, such as crossover and bit mutation. Crossover is applied to two chromosomes (parents) and creates two new chromosomes (offspring) by selecting a random position along the coding and splicing the section that appears before the selected position in the first string with the section that appears after the selected position in the second string and vice versa (see Figure 1). Bit mutation simply offers the chance to flip each bit in the coding of a new solution.

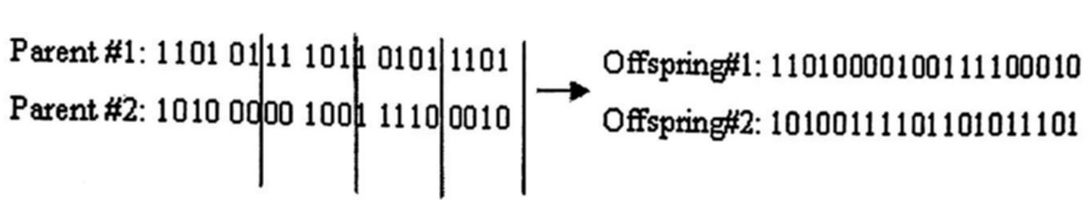
Parent #1: 1101 01|11 101|1 0101|1101
Parent #2: 1010 00|00 100|1 1110|0010
Offspring#1: 11010000100111100010
Offspring#2: 10100111101101011101

Figure 1. *Four-points crossover operators*

The process is halted if a suitable solution has been found or if the available computing time has expired, otherwise, the process proceeds to step 3 where the new chromosomes are scored, and the cycle is repeated.

2.2 Implementation and Empirical Methods.

Mapping Objective Functions to Fitness Form. In many problems, the objective is more naturally stated as the minimization of some cost function *g(x)* rather than the maximization of some utility or profit function *u(x)*. Even if the problem is naturally stated in maximization form, this alone does not guarantee that the utility function will be non negative for all *(x)* as we require in fitness function (a fitness function must be a non negative figure of merit [5]. In normal operations research work, to transform a minimization problem to a maximization problem we simply multiply the cost function by a minus one. In genetic algorithm work, this operation alone is insufficient because the measure thus obtained is not guaranteed to be non negative in all instances. With GAs, the following cost-to-fitness transformation is commonly used:

$$\begin{aligned} f(x) &= C_{\max} - g(x) && \textit{when } g(x) < C_{\max} \\ &= 0 && \textit{otherwise} \end{aligned}$$

$C_{\max}$ may be taken as the largest g value observed thus far. For the problem of nonlinear function in this paper, we take this transformation form.

Fitness Scaling. In order to achieve the best results of GAs, it is necessary to regulate the level of competition among members of the population. This is precisely what we do when we perform fitness scaling. Regulation of the number of copies is especially important in small population genetics algorithms. At the start of GAs runs, it is common to have a few extraordinary individuals in a population of mediocre colleagues. If left to the normal selection rule *(*$pselect_i = f_i / \sum f$*)*, the extraordinary individuals would take over a significant proportion of the finite population in a single generation, and this is undesirable, a leading cause of premature convergence. Later on during a run, we have a very different problem. Late in a run, there may still be significant diversity within the population; however, the population average fitness may be close to the population best fitness. If this situation is left alone, average members and best members get nearly the same number of copies in future generations, and the survival of the fittest necessary for improvement becomes a random walk among the mediocre. In both cases, at the beginning of the run and as the run matures, fitness scaling can help.

Constraints. We deal with the dimension constraints by coding equations and deal with time constraints this way: a genetics algorithm generates a sequence of parameters to be tested using the system model, objective function, and the constraints. We simply run the model, evaluate the objective function, and check to see if any constraints are violated. If not, the parameter set is assigned the fitness value corresponding to the objective function evaluation. If constraints are violated, the solution is infeasible and thus has no fitness.

Codings. When GAs manages a practical problem, the parameters of the problem are always coded into bit strings. In fact, coding designs for a special problem is the key to using GAs effectively. There are two basic principles for designing a GAs coding [6]: (1) The user should select a coding so that short, low order schemata are relevant to the underlying problem and relatively unrelated to schemata over other fixed positions. (2) The user should select the smallest alphabet that permits a natural expression of the problem. Based on the characteristic and structure of nonlinear function, instead of choosing the concatenated, multiparamerted, mapped, fixed-point coding, crossed binary coding is designed according to the two principles above. The coding method of a nonlinear function is as follows: The studied case considers two optimization variables encoded using a binary system, which consists in altering one bit of each variable. Then we place the highest bit of reach local string at the site from 1st to *n*th in nonlinear function chromosome and place the second highest bit of each local string at the site from (*n*+1)th to 2*n*th, and so on. Then we can obtain a nonlinear function chromosome (see Figure 2).

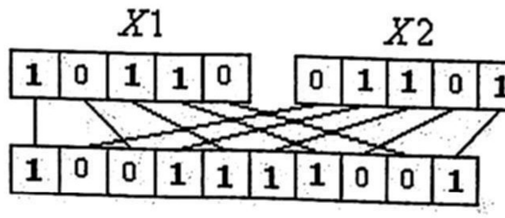

Figure 2. *Illustration of the encoding method for a small size example*

The reason for using crossed binary coding, because this codification is suitable for the continuous and discontinuous variables, and can be analyzed in theory as follows:

- Because of the strong relationship among the parameters, the highest bit in each local string in binary codings determines the basic structure among every parameter, and the second highest bit in each local string determines finer structure among every parameter, and so on for the third, the forth, etc.
- The schema defining length under crossed coding (n) is shorter than the length under concatenated, mapped, fixed-point coding *(nK-K+1)*.

According to the schema theorem: short schemata cannot be disturbed with high frequency, the schema under crossed coding has a greater chance to be reproduced in the next generation. Due to its combining the characteristics of function optimization with schema theorem and successful binary alphabet table, crossed coding demonstrates greater effectiveness than the ordinary coding method in our implementation.

Local string formation is achieved this way: for a parameter $x \in [x_{\min}, x_{\max}]$ that needs to be coded, transform it to a binary coding $X \in [0, 2^K]$ first (appropriate length *K* is determined by the desired degree of precision) and then map it to the specified interval $\left[x_{\min}, x_{\max}\right]$. In this way, the precision of this mapped coding may be calculated as $\delta = \left({x_{\max} - x_{\min}}/{2^{K} - 1} \right)$. In fact, this means that the interval from $x_{\min}$ to $x_{\max}$ is divided into $2^{K} - 1$ parts, because the biggest binary string that has a length of *K* equals the decimal number $2^{0} + 2^{1} + 2^{2} + \ldots + 2^{K-1}$. Then, we can obtain $x = x_{\min} + \delta X$, and a local string for parameter *x* with a length of *K* is obtained.

To illustrate the coding scheme to the size variables more clearly, we also want to give a simple example. For the minimization problem: $\min z = f(x, y)$ in which $x \in [300,700]$ and $y \in [700,1200]$, if we adopt a string length of 5 for each local string and $X: 10110,\ Y: 01101$ is an initial solution, we will get the chromosome 1001110001 (see Figure 3) and obtain:

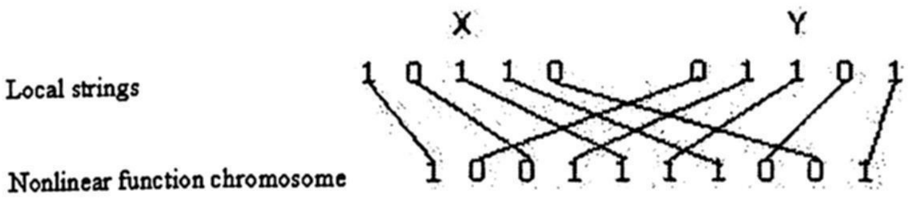

Figure 3. *Multiparameter crossed binary codings*

$$\begin{aligned} x &= x_{\min} + \delta_x X = 300 + \left[{(700-300)}/{(2^5 - 1)} \right] \left(2^4 \times 1 + 2^3 \times 0 + 2^2 \times 1 + 2^1 \times 1 + 2^0 \times 0 \right) \\ &= 300 + \left({400}/{31} \right) \times 22 \\ &= 583.871 \end{aligned}$$

$$\begin{aligned} y &= y_{\min} + \delta_y Y = 700 + \left[{(1200-700)}/{(2^5 - 1)} \right] \left(2^4 \times 0 + 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 \right) \\ &= 700 + \left({500}/{31} \right) \times 13 \\ &= 909.677 \end{aligned}$$

Reproduction. The reproduction operator may be implemented in algorithmic form in a number of ways. In this paper, we take the easiest methods Roulette wheel [6].

Crossover. Crossover operator can take various forms, i.e., one-point crossover, multi-point crossover [7]. It is commonly believed that multi-point crossover has better performance. The number of crossover points in a multi-points crossover operator is determined by the string structure. In this paper, four-points crossover operator is adopted. The crossover rate plays a key role in GAs implementation. Different values for crossover rate ranging from 0.4 to 1.0 were tried, and the results demonstrate that the values ranging from 0.6 to 0.95. In this paper, we take 0.6 as a crossover rate.

Mutation operation. After selection and crossover, mutation is then applied on the resulting population, with a fixed mutation rate. The number of individuals on which the mutation procedure is carried out is equal to the integer part of the value of the population size multiplied by the mutation rate. These individuals are chosen randomly among the population and then the procedure is applied .The mutation rate using in this paper is 0.40.

Elitism. The elitism consists in keeping the best individual from the current population to the next one. In this paper, we take 1 as elitism value.

2.3 Population-Related Factors.

Population Size. The GAs performance is influenced heavily by population size. Various values ranging from 20 to 200 population size were tested. Small populations run the risk of seriously under covering the solution space, a small population size causes the GAs to quickly converge on a local minimum, because it insufficiently samples the parameter space, while large populations incur severe computational penalties. According to our experience, a population size range from 50 to 200 is enough our problem. In this paper and according to our experience, we take 200 as a population size.

Initial Population. It is demonstrated that a high-quality initial value obtained from another heuristic technique can help GAs find better solutions rather more quickly than it can from a random start. However, there is possible disadvantage in that the chance of premature convergence may be increased. In this paper, the initial population is simply chosen by random.

Termination Criteria. It should be pointed out that there are no general termination criteria for GAs. Several heuristic criteria are employed in GAs, i.e., computing time (number of generations), no improvement for search process, or comparing the fitness of the best-so-far solution with average fitness of all the solutions. All types of termination criteria above were tried; the criteria of computing time are proven to be simple and efficient in our problem. In our experience, 200-1000 generations simulation is enough for a complicated problem as our problem. The best results were obtained when the numbers of generations were taken as 1000 for our problem. However, we need to stress that the Genetics Algorithms parameters for traditional GAs and GAs with crossed binary coding are the same. As shown in Table 1 presents the Genetic Algorithms parameters used in this study.

Table.1 Genetic algorithms parameters

Population size	200
Generation number	1000
Survival rate	0.6
Mutation rate	0.4
Elitism	1

3 Examples and Analysis

Four problems are given here to demonstrate the effectiveness of GAs. The mathematical functions are composed of four problems with three objective functions, each one depending on two independent variables (*X1* and *X2*), the choice of these mathematical functions is due to their particular behavior [8]. The results are presented in Table. 1

Table. 1 Results founded by traditional GAs and GAs with crossed binary coding

Functions	Variable X1	Variable X2	Minimum	CPU time(s)*	
				GAs[(a)]	GAs[(b)]
Problem 1					
$f = \frac{(x_1-2)^2}{2} + \frac{(x_2+1)^2}{13} + 3$	2	-1	3	10	<1
$f_2 = \frac{(x_1+x_2-3)^2}{36} + \frac{(-x_1+x_2+2)^2}{8} - 17$	2.5	0.5	17	13	<1
$f_3 = \frac{(3x_1-2x_2-1)^2}{175} + \frac{(-x_1+2x_2)^2}{17} - 13$	0.5	0.25	-13	17	<1
$x = (x_1, x_2) \in [-4,4]^2$					
Problem 2					
$f_1 = \frac{(x_1-2)^2}{2} + \frac{(x_2+1)^2}{13} + 3$	2	-1	3	10	<1
$f_2 = \frac{(x_1+x_2-3)^2}{36} + \frac{(-x_1+x_2+2)^2}{8} - 17$	2.5	0.5	-17	13	<1
$f_3 = \frac{(3x_1-2x_2+4)^2}{18} + \frac{(x_1-x_2+1)^2}{27} + 15$	-2	-1	15	20	<1
$x = (x_1, x_2) \in [-4,4]^2$					
Problem 3					
$f_1 = \frac{x_1^2}{2} + \frac{(x_2+1)^2}{13} + 3$	0	-1	3	10	<1
$f_2 = \frac{x_1^2}{2} + \frac{(2x_2+2)^2}{15} + 1$	0	-1	1	10	<1
$f_3 = \frac{(x_1+2x_2-1)^2}{175} + \frac{(2x_2-x_1)^2}{27} - 13$	0.5	0.25	-13	15	<1
$x = (x_1, x_2) \in [-4,4]^2$					
Problem 4					
$f_1 = 0{,}5(x_1^2+x_2^2) + \sin(x_1^2+x_2^2)$	0	0	0	10	<1
$f_2 = \frac{(3x_1-2x_2+4)^2}{8} + \frac{(x_1-x_2+1)^2}{27} + 15$	-2	-1	15	20	<1
$f_3 = \frac{1}{(x_1^2+x_2^2+1)} - 1{,}1\exp(-x_1^2-x_2^2)$	0	0	-0.1	25	<1
$x = (x_1, x_2) \in [-4,4]^2$					

**CPU time was calculated to this method on Microsoft Windows XP Profesional Intel(R)D CPU 2.80 Ghz, 2.99 GB of RAM.*
GAs[(a)]: Traditional GAs
GAs[(b)]: GAs with crossed binary coding.

From these results, we can see that better results are obtained in comparison with the traditional Genetic Algorithms. In addition, GAs with crossed binary coding results in a faster convergence and the computing time is less than that of traditional GAs. This demonstrates the effectiveness of GAs with crossed binary coding in solving the complicated problem quickly.

Now, several works about some important aspects in our implication of GAs and some problems in practice.

The most important of all is the method of coding. Because of the characteristics and inner structure of the nonlineal function, the commonly adopted concatenated, multiparamerted, mapped, fixed point coding is not effective in searching for the global optimum as soon as possible. However, as is evident from the results of application, crossed binary coding method is well fit for the proposed problem.
Another aspect that affects the effectiveness of genetic procedure considerably is crossover. Corresponding to the proposed coding method, we adopted a four point crossover method. It is commonly believed that multi-point crossover is more effective than the traditional one point crossover method. Nevertheless, we find that it is not the case that the more points to crossover, the better. It is also important to note that the selection of crossover points as well as the way to carry out the crossover should take in account the bit string structure, as is the case in our implication. Despite the demonstrated advantages of GAs algorithms, the feeling persists that there is much to learn about effectively implementing a genetic algorithms. One problem in practice is the premature loss of diversity in the population, which results in premature convergence. Because premature convergence is so often the case in the implementation of GAs according to our computation experience. Something has to be done to prevent it. Our experience makes it clear that the elitism parameter could solve the premature problem effectively and conveniently.

4 Conclusions

Genetics Algorithms are applied to minimize nonlineal function. Satisfactory results are obtained. The obtained experimental results showed that the performance of the GA depends on the codification chosen. Moreover; crossed binary coding method is the best schema, with this codification GAs converges faster and the computing time is less than that of traditional GAs. However as it does not seem easy to envisage a method to select in advance the best schema for a given problem instance, in principle the only way is trying various schemas at the same time and take the value provided for the best one.

Acknowledgements

Youness El Hamzaoui expresses his gratitude to the Consejo Nacional de Ciencia y Tecnología (CONACYT) for the scholarship awarded to postgraduate Ph.D. studies.

References

1. Wang, C., Quan, H., Xu, X., Optimal design of multiproduct batch chemical process using genetic algorithm. Industrial engineering and chemistry research, 1996, 35, 10, 3560-6.
2. El Hamzaoui, Youness; Hernandez, J. A.; Cruz-Chavez, M A; and Bassam, A (2010) "Search for Optimal Design of Multiproduct Batch Plants under Uncertain Demand using Gaussian Process Modeling Solved by Heuristics Methods," Chemical Product and Process Modeling: Vol. 5 : Iss. 1, Article 8.
3. Patel, A.N.,; Mah, R.S.H.; Karimi, I.A. Preliminary design of multiproduct non-continuous plants using simulating annealing. Comput.Chem. Eng.1991, 15, 451.
4. Çelebi, Mehmed(2009) 'A new approach for the genetic algorithm', Journal of Statistical Computation and Simulation, 79: 3, 275 — 297.
5. Holland, J.H., Adaptation in Natural and Artificial Systems. University of
6. Michigan Press Inc., 1975.
7. Goldberg, D.E., Genetic algorithms in search optimization and machine learning.
8. Addison Wesley Publishing Company Inc. 1989.
9. Frantz, D.R., Non-linearities in genetic adaptive search. Academic Press Inc., 1994.
10. Viennet, R. (1997). Nouvel outil de planification experimentale pour l'optimisation multicritere des procedes. These de doctorat, INP Lorraine, France.