

Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis

F. HERRERA, M. LOZANO and J.L. VERDEGAY*

*Department of Computer Science and Artificial Intelligence
University of Granada, Spain*

(Received ; Accepted in final form)

Abstract. Genetic algorithms play a significant role, as search techniques for handling complex spaces, in many fields such as artificial intelligence, engineering, robotic, etc. Genetic algorithms are based on the underlying genetic process in biological organisms and on the natural evolution principles of populations. These algorithms process a population of chromosomes, which represent search space solutions, with three operations: selection, crossover and mutation.

Under its initial formulation, the search space solutions are coded using the binary alphabet. However, the good properties related with these algorithms do not stem from the use of this alphabet; other coding types have been considered for the representation issue, such as real coding, which would seem particularly natural when tackling optimization problems of parameters with variables in continuous domains. In this paper we review the features of real-coded genetic algorithms. Different models of genetic operators and some mechanisms available for studying the behaviour of this type of genetic algorithms are revised and compared.

Key words: genetic algorithms, real coding, continuous search spaces

Abbreviations: GAs – genetic algorithms; BCGA – binary-coded genetic algorithm; RCGA – real-coded genetic algorithm

1. Introduction

Genetic algorithms (GAs) are general purpose search algorithms which use principles inspired by natural genetic populations to evolve solutions to problems (Holland, 1975; Goldberg, 1989a). The basic idea is to maintain a population of chromosomes, which represent candidate solutions to the concrete problem, that evolves over time through a process of competition and controlled variation. Each chromosome in the population has an associated *fitness* to determine which chromosomes are used to form new ones in the competition process, which is called *selection*. The new ones are created using genetic operators such as *crossover* and *mutation*. GAs have had a great measure of success in search and optimization problems. The reason for a great part of their success is their ability to exploit the information accumulated about an initially unknown search space in order to bias subsequent searches into useful subspaces, i.e., *their adaptation*. This is their key

* This research has been supported by DGICYT PB92-0933.

feature, particularly in large, complex, and poorly understood search spaces, where classical search tools (enumerative, heuristic,...) are inappropriate, offering a valid approach to problems requiring efficient and effective search techniques.

Fixed-length and binary coded strings for the representation solution have dominated GA research since there are theoretical results that show them to be the most appropriate ones (Goldberg, 1991a), and as they are amenable to simple implementation. But the GA's good properties do not stem from the use of bit strings (Antonisse, 1989; Radcliffe, 1992). For this reason, the path has been lain toward the use of non-binary representations more adequate for each particular application problem. One of the most important ones is the *real number representations*, which would seem particularly natural when optimization problems with variables in continuous search spaces are tackled. So a chromosome is a vector of floating point numbers whose size is kept the same as the length of the vector, which is the solution to the problem. GAs based on real number representation are called *real-coded GAs* (RCGAs). The use of real coding initially appears in specific applications, such as in (Lucasius et al., 1989) for chemometric problems, and in (Davis, 1989) for the use of metaoperators in order to find the most adequate parameters for a standard GA. Subsequently, RCGAs have been mainly used for numerical optimization on continuous domains (Wright, 1991; Davis, 1991; Michalewicz, 1992; Eshelman et al., 1993; Mühlenbein et al., 1993; Herrera et al., 1994; Herrera et al., 1995). Until 1991 specific theoretical studies about RCGA operation weren't done and so the use of these algorithms was controversial; researchers familiar with fundamental GA theory didn't understand the great success of RCGAs since this suggested that binary coding should be more effective than codings based on large alphabets. Later, tools for the theoretical treatment of RCGAs were proposed (Wright, 1991; Goldberg, 1991a; Radcliffe, 1991a; Eshelman et al., 1993) and so their power was corroborated.

The main objective of this paper is to deal with the RCGAs. To do that, first we study the binary coding and its advantages and drawbacks. Then we study the main issues related with RCGAs and the tools for the analysis of the RCGAs.

We need to highlight that this paper is advisable and interesting for people working in GAs and people that need a power search procedure for problems with large, complex, and poorly understood search spaces. RCGAs demonstrate to be one of the most appropriate search methods on problems with these features where continuous variables are implied.

We set up the paper as follows. In Section 2 we attempt to describe the principal aspects of GAs, thinking of people that are not familiar-

ized with them. Then, in Section 3 we expose the particular features of binary-coded GAs (BCGAs) and show the reasons argued for preferring binary alphabet, which has been the most used through GA history. We also point out the principal problems that appear when BCGAs are applied. In Section 4 we attempt the RCGAs, we expose their advantages, we present and compare the crossover and mutation operators proposed for these algorithms in the literature, we deal with the application of RCGAs for handling convex spaces, and finally, we treat the hybridization of RCGAs with other search methods. In Section 5 we report tools that allow the behaviour of RCGAs to be studied from a theoretical point of view. In Section 6 some conclusions are pointed out.

2. Overview of GAs

A GA starts off with a population of randomly generated *chromosomes*, and advances toward better chromosomes by applying genetic operators, modeled on the genetic processes occurring in nature. The population undergoes evolution in a form of natural selection. During successive iterations, called *generations*, chromosomes in the population are rated for their adaptation as solutions, and on the basis of these evaluations, a new population of chromosomes is formed using a selection mechanism and specific genetic operators such as crossover and mutation. An *evaluation* or *fitness function*, f , must be devised for each problem to be solved. Given a particular chromosome, a solution, the fitness function returns a single numerical fitness, which is supposed to be proportional to the utility or adaptation of the solution which that chromosome represents.

Next, we offer a more detailed description of GAs by considering the following themes: the structure of a basic GA, the representation issue, the selection mechanism, the recombination through crossover and mutation operators and the GAs applications.

2.1. STRUCTURE OF A GA

Although there are many possible variants of the basic GA, the fundamental underlying mechanism operates on a population of chromosomes or individuals, which representing possible solutions to the problem, and consists of three operations:

1. evaluation of individual fitness,
2. formation of a gene pool (intermediate population) through selection mechanism and

Figure 1. Structure of a GA

```

Procedure Genetic Algorithm
begin (1)
   $t = 0$ ;
  initialize  $P(t)$ ;
  evaluate  $P(t)$ ;
  While (Not termination-condition) do
    begin (2)
       $t = t + 1$ ;
      select  $P(t)$  from  $P(t - 1)$ ;
      recombine  $P(t)$ ;
      evaluate  $P(t)$ ;
    end (2)
end (1)

```

3. recombination through crossover and mutation operators.

Figure 1 shows the structure of a basic GA. $P(t)$ denotes the population at generation t .

2.2. REPRESENTATION ISSUE

Representation is a key issue in GA work because GAs directly manipulate a coded representation of the problem and because the representation schema can severely limit the window by which a system observes its world (Koza, 1992). Fixed-length and binary coded strings for the representation solution have dominated GA research since there are theoretical results that show them to be the most effective ones (Goldberg, 1991a), and as they are amenable to simple implementation. But the GA's good properties do not stem from the use of bit strings (Antonisse, 1989; Radcliffe, 1992). This reason motivated in many cases that GA practitioners devised non-binary representations, accompanied of genetic operators, more natural for the specific application problems. Examples of such cases are the following:

- *vectors of floating point numbers*, for chemometric problems (Lucasius et al., 1989), numerical function optimization (Davis, 1991; Wright, 1991; Michalewicz, 1992; Herrera et al., 1994), evolving rule sets for classification (Michielssen et al., 1992), optimal multilayer filter design (Corcoran et al., 1994), tuning fuzzy logic controllers (Herrera et al., 1995), etc.

- *vectors of integer numbers*, for function optimization (Bramlette, 1991), parametric design of aircraft (Bramlette et al., 1991), unsupervised learning of neural networks (Ichikawa et al., 1993), etc.
- *ordered lists*, for schedule optimization problems (Syswerda, 1991), traveling salesman problem (Whitley et al., 1989), job-shop scheduling problems (Fox et al., 1991)
- *lisp expressions*, for evolving computer programs for control tasks, robotic planning and symbolic regression (Koza, 1992).
- *two-dimensional matrix of integer numbers*, for the linear transportation problem (Michalewicz, 1991).

2.3. SELECTION MECHANISM

Let's consider P a population with chromosomes C_1, \dots, C_N . The selection mechanism produces a intermediate population, P' , with copies of chromosomes in P . The number of copies received for each chromosome depends on its fitness; chromosomes with higher fitness usually have a greater chance of contributing copies to P' . The selection mechanism consists of two steps, the selection probability calculation and the sampling algorithm.

2.3.1. Selection Probability Calculation

For each chromosome C_i in P the probability, $p_s(C_i)$, of including a copy of such chromosome into P' is calculated. Most well-known selection mechanisms use the *proportional selection* (Holland, 1975; Goldberg, 1989a) where $p_s(C_i)$, $i = 1, \dots, N$, is calculated as

$$p_s(C_i) = \frac{f(C_i)}{\sum_{j=1}^N f(C_j)}.$$

In this way, chromosomes with above-average fitness tend to receive more copies than those with below-average fitness.

A different approach is the *ranking selection* (Baker, 1985); the chromosomes are sorted in order of raw fitness, and then $p_s(C_i)$, $i = 1, \dots, N$, is computed according to the rank of C_i by using a non-increasing assignment function.

2.3.2. Sampling Algorithm

It reproduces, based on the selection probabilities computed before, copies of chromosomes to form P' . The classical one is the *stochastic*

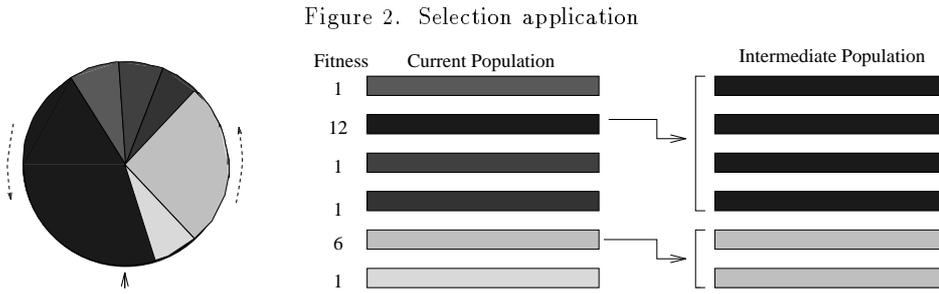
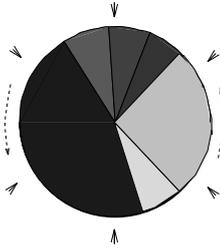


Figure 3. Stochastic universal sampling



sampling with replacement (Holland, 1975; Goldberg, 1989a); the population is mapped onto a roulette wheel, where each chromosome C_i is represented by a space that proportionally corresponds to $p_s(C_i)$. By repeatedly spinning the roulette wheel, chromosomes are chosen until all available positions in P' are filled. Figure 2 shows an example of the application of a selection mechanism based on *stochastic sampling with replacement*.

One of the most efficient sampling algorithms is the *stochastic universal sampling* (Baker, 1987). This procedure guarantees that the number of copies of any chromosome, C_i , is bounded by the floor and by the ceiling of its *expected number of copies*, i.e., $p_s(C_i) \cdot N$. This algorithm simulates a roulette wheel with N equally distributed pointers. A single spin of the wheel determines the number of copies assigned to every chromosome C_i , which corresponds with the number of pointers that point to the region associated with C_i . Figure 3 shows the roulette wheel associated with the selection process of Figure 2 when *stochastic universal sampling* is used.

A wide set of selection mechanisms are reviewed in (Bäck et al., 1991b).

2.4. RECOMBINATION THROUGH CROSSOVER AND MUTATION

After selection has been carried out, the construction of the intermediate population is complete and the operators of crossover and mutation are applied.

2.4.1. *Crossover Operator*

The crossover operator is a method for sharing information between chromosomes; it combines the features of two parent chromosomes to form two offspring, with the possibility that good chromosomes may generate better ones. The crossover operator is not usually applied to all pairs of chromosomes in the intermediate population. A random choice is made, where the likelihood of crossover being applied depends on probability defined by a crossover rate, the crossover probability, p_c . The crossover operator plays a central role in GAs, in fact it may be considered to be one of the algorithm's defining characteristics, and it is one of the components to be borne in mind to improve the GA behaviour (Liepins et al., 1992). Definitions for this operator (and the next one) are very dependent on the particular representation chosen.

2.4.2. *Mutation Operator*

The mutation operator arbitrarily alters one or more components, *genes*, of a selected chromosome so as to increase the structural variability of the population. The role of mutation in GAs is that of restoring lost or unexplored genetic material into the population to prevent the premature convergence of GA to suboptimal solutions; it insures that the probability of reaching any point in the search space is never zero. Each position of every chromosome in the population undergoes a random change according to a probability defined by a mutation rate, the mutation probability, p_m .

We should point out that after crossover and mutation, an additional selection strategy, called *elitist strategy*, may be adopted (De Jong, 1975): to make sure that the best performing chromosome always survives intact from one generation to the next. This is necessary since it is possible that the best chromosome disappears, thanks to crossover or mutation.

2.5. APPLICATIONS OF GAS

GAs may deal successfully with a wide range of problem areas. Mainly, the reasons for this success are (Goldberg, 1991c): 1) GAs can solve hard problems quickly and reliably, 2) GAs are easy to interface to existing simulations and models, 3) GAs are extensible and 4) GAs are easy to hybridize. All these reasons may be summed up in only one:

GAs are *robust*. GAs are more powerful in difficult environments where the space usually is large, discontinuous, complex and poorly understood. They are not guaranteed to find the global optimum solution to a problem, but they are generally good at finding acceptably good solutions to problems acceptably quickly.

In (Goldberg, 1989a) a wide review of applications realized before 1989 may be looked. During last years, GA applications have enormously grown in many fields: engineering (Goldberg, 1989a; Davis, 1991), numerical function and combinatorial optimization (Goldberg, 1989a; Michalewicz, 1992; Davis, 1991), robotic (Davidor, 1991), classifier systems (Holland et al., 1986; Booker et al., 1989; Belew et al., 1991; Forrest, 1993), learning (Grefenstette, 1990), pattern recognition (Forrest et al., 1993), neuronal networks (Whitley et al., 1992; Forrest, 1993), fuzzy systems (Cordon et al., 1995), artificial life (Belew et al., 1991), etc.

3. Binary-Coded Genetic Algorithms

This section is devoted to BCGAs. First we present the binary coding. We explain the crossover and mutation operators for this type of coding and show a simple example of BCGA iteration. The schema theory, that describes the GA behaviour, is treated. Then we expound the arguments for using the binary alphabet, and finally, some drawback of BCGAs, due to the binary coding use, are presented.

3.1. BINARY CODING

To represent the elements of a search space $S = S_1 \times \dots \times S_n$ by means of the binary alphabet, a function $cod_i : S_i \rightarrow \{0, 1\}^{L_i}$, $L_i \in N$, should be specified, which codes each element in S_i using binary strings of length L_i .

An element $x = (x_1, \dots, x_n) \in S$ ($x_i \in S_i$) is represented by linking together the codings of each one of its components $cod(x) = cod_1(x_1) \dots cod_n(x_n)$.

3.2. CROSSOVER OPERATOR

The classical crossover operator is the *simple crossover* (Holland, 1975; Goldberg, 1989a), in which, given two chromosomes $C_1 = (c_1^1 \dots c_L^1)$ and $C_2 = (c_1^2 \dots c_L^2)$ the offspring $H_1 = (c_1^1, \dots, c_i^1, c_{i+1}^2, \dots, c_L^2)$ and $H_2 = (c_1^2, \dots, c_i^2, c_{i+1}^1, \dots, c_L^1)$ are generated, where i is a random number belonging to $\{1, \dots, L-1\}$. Figure 4 shows an example of this operator's application.

Figure 4. Simple crossover

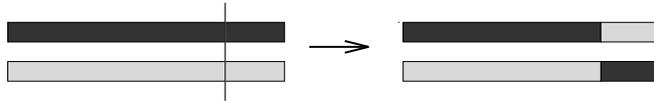
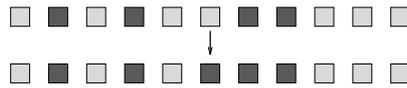


Figure 5. Mutation



Numerous research papers have been directed at finding alternative more powerful crossover operators for BCGAs. Two of the most important ones are:

- *n-point crossover* (Eshelman et al., 1989): this is a generalization of the simple crossover; n crossover points are randomly selected and the segments of the parents, defined by them, are exchanged for generating the offspring.
- *uniform crossover* (Syswerda, 1989): the values of each gene in the offspring are determined by the uniform random choice of the values of this gene in the parents.

Other types of crossover operators under binary coding are reported in (Eshelman et al., 1989).

3.3. MUTATION OPERATOR

Given a chromosome, a gene is randomly chosen and its value is swapped; "1" for "0" and viceversa (Holland, 1975; Goldberg, 1989a). Figure 5 shows an example of the application of mutation operators.

3.4. EXAMPLE OF BCGA ITERATION

Next, we show an example of an iteration of a BCGA during only one generation for the function $f(x) = x^2$. The selection operator is the proportional selection along with the stochastic sampling with replacement. The control GA parameters are $p_c = 0.5$, $p_m = 0.001$ and the population size is 4. Given the low value of p_m , no mutation is performed. The coding is based on the binary code.

EXAMPLE 1. Table I shows the GA execution during a generation, t . This generation may be divided into two step: the selection mechanism

Table I. Example of a BCGA iteration

Selection Mechanism				Recombination		
Population	Fitness	$\frac{f(C_i)}{\sum_{j=1}^N f(C_j)}$	Copies	Population	New	Fitness
C_i	$f(C_i)$		Obtained	Intermediate	Population	
0110	36	0.19	1	01 10	0101	25
0010	4	0.02	0	10 01	1010	100
1001	81	0.43	2	1001	1001	81
1000	64	0.34	1	1000	1000	64
Worst	4					25
Average	46.25					67.5
Best	81					100

application and the recombination process. We should point out that column 4 shows the number of copies assigned to each chromosome by the selection mechanism and that in column 5 the "|" symbol indicates the crossover point chosen for the crossover operation.

A simple generation is sufficient for understanding the GA power; Worst, Average and Best measures have grown during this generation.

3.5. THE SCHEMA THEORY

One of the advantages of BCGAs is that their simple formulation is amenable for a theoretical study. Holland in (Holland, 1975) developed the *schema theory*, which presents results that explain the good GA behaviour.

Next we define the concept of *schema* in order to present the most important result of this theory: the *schema theorem*.

3.5.1. Schema Concept

GAs direct the search in the search space in a global form. The main concept that describes this orientation is that of *schema*. A schema is a similarity pattern that describes a subset of strings with similar features in some positions. Given an alphabet \mathcal{A} , a schema is a string defined over the alphabet $\mathcal{A} \cup \{\star\}$. The symbol \star is a *don't care* that may be used as a substitute for any symbol in \mathcal{A} . For example, if $\mathcal{A} = \{0, 1\}$ then the schema $\epsilon = \star 11 \star 010$ represents the chromosomes:

$$\{(0110010), (1110010), (0111010), (1111010)\}.$$

Two useful features of a schema ϵ are the following:

- Order of ϵ , denoted by $o(\epsilon)$: the number of fixed symbols in ϵ .
- Defining length of ϵ , denoted by $\delta(\epsilon)$: the difference between the first and the last fixed symbol in ϵ .

Now, we present the *schema theorem*, which assures that BCGAs allocate their fitness function evaluations in an intelligent way.

3.5.2. Schema Theorem

Let us suppose a binary coding with L being the length of the chromosomes, $f(\epsilon)$ the average fitness of the instances of ϵ in the population, \bar{f} the average fitness of the elements in the population and $m(\epsilon, t)$ the number of instances of ϵ (chromosomes that are contained in ϵ) in the population at generation t . The expected number of instances of this schema at the next generation, $t+1$, after applying the selection, simple crossover and mutation operators, obeys the next expression (Holland, 1975):

$$m(\epsilon, t+1) \geq m(\epsilon, t) \cdot \frac{f(\epsilon)}{\bar{f}} \cdot \left(1 - p_c \cdot \frac{\delta(\epsilon)}{L-1}\right) \cdot (1 - p_m)^{o(\epsilon)}.$$

$m(\epsilon, t) \cdot \frac{f(\epsilon)}{\bar{f}}$ is the expected number of instances of ϵ after the selection mechanism application. $\left(1 - p_c \cdot \frac{\delta(\epsilon)}{L-1}\right)$ is approximately the probability of survival of ϵ after the crossover operator application, it is high when $\delta(\epsilon)$ is low. $(1 - p_m)^{o(\epsilon)}$ is the probability of survival of ϵ after mutation operator application, it is high when $o(\epsilon)$ is low. This result motivates the *schema theorem* (Holland, 1975):

THEOREM 1. *Short, low-order, above-average schemata, called building blocks, receive exponentially increasing trials in subsequent generations.*

An immediate result of this theorem is the *building block hypothesis* (Goldberg, 1989a):

RESULT 1. *A BCGA seeks near-optimal performance through the juxtaposition of building blocks.*

This result means that the recombination of the building blocks kept in the population allows chromosomes near the optimum to be reached. This is a GA's key feature, since it predicts good behaviour for these types of algorithms. Example 2 illustrates the schema theorem in detail for the BCGA iteration in Example 1.

Table II. Schemata processing

ϵ	$m(\epsilon, t)$	$f(\epsilon)$	$m(\epsilon, t+1)$	$f(\epsilon)$
1***	2	72.5	3+	81.66
01**	1	36	1	25
1**1	1	81	1	81
0**0	2	20	0-	

EXAMPLE 2. Table II shows the effect of the generation t on four interesting schemata. These schemata are different according their order, length and performance (average fitness of instances). Column 2 shows the number of instances of schemata before genetic process is applied and column 3 shows the schema average fitness of schemata in this stage. The following columns represent the same information but after applying selection and crossover. A plus sign in column 4 indicates that the generation has caused the number of instances of schema to increase as compared to the number of instances shown in column 2. The minus sign is defined in the same way.

1*** is a short ($\delta(1***) = 0$), low-order ($o(1***) = 1$) and above-average schema ($f(1***) = 72.5$ and $\bar{f} = 46.25$), so it has received the exponentially increasing of instances predicted by the schema theorem. 01** is a below-average schema that keep the same number of instance because it is short and low-order; the effect of crossover is not significant if schemata show these features. 1**1 may be easily disrupted i.e., it is amenable to lose instances due crossover, however it might keep its number of instances since it is above-average. The case of 0**0 is different, this schema has lost its instances.

3.6. ARGUMENTS FOR USING THE BINARY ALPHABET

There have mainly been two arguments for using the binary alphabet. First, this alphabet maximizes the level of *implicit parallelism* (a GA's essential property) (Goldberg, 1991a). Later, we shall see that there are a number of counter-arguments to this. Second, the alphabetic characters associated with a higher cardinal alphabet are not as well represented in a finite population which forces the use of larger population sizes (Reeves, 1993) and thus efficiency may decrease. This is a great problem when the evaluation of the fitness function is very expensive. So, in this cases, low cardinal alphabets have to be used.

3.6.1. Binary Coding Favours on the Implicit Parallelism Property

An important interpretation of the schema theorem is suggested in (Holland, 1975): GAs process schemata instead of chromosomes. Al-

Table III. Binary and octal coded elements

Binary	Octal	Fitness
000	0	22
011	3	8
101	5	11
111	7	3

though GAs superficially seem to process only the particular individual binary chromosomes actually present in the current population, they actually processes, in *parallel*, a large amount of useful information concerning unseen schemata. This important property is called *implicit parallelism* and is one of the most important underlying fundamentals of the genetic search. The following results associate the maximum level of implicit parallelism with the binary alphabet (Goldberg, 1991a):

THEOREM 2. *For a given information content, strings coded with smaller alphabets are representatives of larger numbers of similarity subsets (schemata) than strings coded with larger alphabets.*

An important result derived from this theorem is Result 2.

RESULT 2. *The binary alphabet offers the maximum number of schemata per bit of information.*

Since implicit parallelism property means that GAs process schemata rather than individual chromosomes, and according to Result 2, it is possible to deduce that the binary alphabet maximizes the implicit parallelism level, because it allows the sampling of the maximum number of schemata per chromosome in the population. Therefore, the maximum efficiency level is achieved. The influence of this property on the search process may be observed in the next example (Goldberg, 1991a).

EXAMPLE 3. Let us suppose that four elements were coded using the binary and octal alphabets, as is shown in Table III. When looking at the octal coded elements, no relation between these elements and their fitness may be induced. However, when analyzing the binary coding, each element belongs to different subsets of elements related to it through similarities in the values of specific positions. Common properties of the promising elements may be found through these similarities. In particular, we would be able to explain the goodness of the first element (000) since it has 00 on its left side, or 0 in the medium position,

as happens in element 101. The BCGA shall recombine the similarities (schemata) of the elements with the best fitness for producing better elements.

Example 3 offers a clearly explanation of why binary coding favours on the implicit parallelism property. However, there are different arguments against this fact that will be presented in Subsection 3.7.3.

3.6.2. Low Cardinal Alphabets

There are many problems in which the number of fitness evaluations is limited. Therefore, the use of small population sizes is required. In (Reeves, 1993) is attempted the specification of the minimum population size to ensure an *initial* population (generated at random) with the suitable property to avoid subsequent poor behaviour. For Reeves this property lies in the following principle:

"Every possible point in the search space should be reachable from the initial population by crossover only"

Given an alphabet \mathcal{A} ($|\mathcal{A}| = q$), a population size N and a chromosome length L , this principle may only be achieved if at least for each position i ($i = 1, \dots, L$) and for each symbol $s \in \mathcal{A}$ there is a chromosome in the initial population such that i has s . Reeves calculated the probability, $p^*(q, N, L)$, that this happens, as follows:

$$p^*(q, N, L) = \left(\frac{q!S(N, q)}{q^N} \right)^L,$$

where $S(M, q)$ is the Stirling number of the second kind, which is generated by the recursion

$$S(N+1, q) = S(N, q-1) + qS(N, q), \quad N \geq 1, \quad q \geq 2 \quad \text{and} \quad S(N, 1) = 1, \quad \forall N.$$

If a particular bound, α , for the previous probability is fixed, the relationship between q and N may be studied. More specifically, given a value for q , the minimum value of N that allows the bound α to be kept may be calculated. In this way, for any alphabet we may determinate the minimum population size needed for keeping a determinate level of efficiency represented by α . For different values of L , 20, 40, ..., 200, and different values of q , 2, 3, ..., 8, the minimum value of N , N_{min} , such as $P^*(q, N_{min}, L) > \alpha$, was determinated by Reeves, with $\alpha = 0.95, 0.99$ and 0.999. The results showed that, the higher q is the higher N_{min} has to be for keeping the same bound of p^* .

The main conclusion of this is that for a given problem, the GA's computational effort using an alphabet with $q > 2$ (power of two) for

obtaining a particular solution shall be higher than using $q = 2$, since a greater population size should be kept. This is intuitively reasonable, for example, eight chromosomes are needed in order to represent all the possible values of an alphabet with $q = 8$ in a position i , whereas if $q = 2$, with the combination of only two strings (000 and 111), the same information is represented, since the crossover operator shall be able to generate values that are not explicitly represented in the initial population.

In (Tate et al., 1993) a measure called *expected allele coverage* was defined with the same function as p^* , i.e., given an alphabet \mathcal{A} ($|\mathcal{A}| = q$), to determine the degree in which a population size, N , is adequate. Studies based on this measure showed that complex codings may require larger population sizes than the binary coding for achieving a desired expected allele coverage. Clearly, this result is like to the one obtained by Reeves. For compensating the problems associated with poor expected allele coverages Tate et al. proposed to adjust the mutation probability. They showed that problems requiring non-binary coding may benefit from mutation probabilities much higher than those generally used with binary coding, which are very small (Goldberg, 1989a).

3.7. DRAWBACKS OF BINARY CODING

The binary representation meets certain difficulties when dealing with *continuous search spaces* with large dimensions and a great numerical precision is required. A problem occurs when a variable may only have a finite number (which is not a power of 2) of discrete valid values. In this case, some of the binary codes are *redundant*. In the following these two issues are considered together with different arguments against the binary coding favours on the implicit parallelism property expounded in the Subsection 3.6.1.

3.7.1. Problems in Continuous Search Spaces

Two types of binary coding were mainly considered for representing a parameter x_i belonging to a continuous interval $S_i = [a_i, b_i]$: *the binary code* (Holland, 1975; Goldberg, 1989a) and *the Gray code* (Caruana et al., 1988).

Prior to the codification, a transformation from the interval $[a_i, b_i]$ to the set $\{0, \dots, 2^{L_i}\}$ (L_i is the number of bits in the coding) is carried out. Then, the resultant elements are coded using one of the aforesaid codings. This transformation implies a discretization of the interval $[a_i, b_i]$, which has the following associate precision, ρ :

$$\rho = \frac{b_i - a_i}{2^{L_i} - 1}.$$

A string (c_1, \dots, c_{L_i}) , $c_i \in \{0, 1\}$, coded under the binary code represents the integer $v \in \{0, \dots, 2^{L_i} - 1\}$,

$$v = \sum_{j=1}^{L_i} c_j 2^{(L_i-j)}.$$

The Hamming distances of the Gray codings of two consecutive integers differ in one. Given a binary coded string (c_1, \dots, c_{L_i}) the conversion formula for obtaining the corresponding Gray coded string (g_1, \dots, g_{L_i}) is the following:

$$g_k = \begin{cases} c_1 & \text{if } k = 1 \\ c_{k+1} \oplus c_k & \text{if } k > 1 \end{cases}$$

where \oplus denotes addition module 2. The inverse conversion is defined by means of the following expression.

$$c_k = \sum_{j=1}^k g_j,$$

where the sum is done in module 2. Table IV shows the coding of integers 1 to 15 using binary and Gray codings.

The L_i parameter determines the search space magnitude. Also, it delimits the precision of the returned solution (ρ depends on L_i). This may produce difficulties when problems with large dimensions and requirements of great numerical precision are dealt with. In (Michalewicz, 1992) an example of such a problem was reported:

EXAMPLE 4. For 100 variables with domains in the range $[-500, 500]$, where the precision of six digits after the decimal point is required, the length of the binary solution vector is 3000. This, in turn, generates a search space of about 10^{1000} .

The BCGA's performance in these problems may be poor (Schraudolph et al., 1992). During the first stages, the algorithm wastes great efforts evaluating the less significant digits of the binary coded parameters. However, their optimum values shall depend on the most significant digits, therefore, whereas these ones don't converge, their manipulation shall be useless. When convergence of the most significant digits

Table IV. Comparison of binary and Gray codings

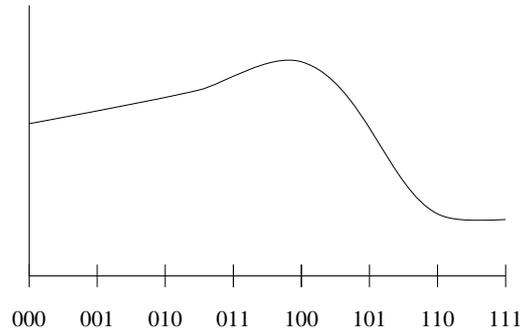
Integers	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

is achieved, it is not necessary to waste more efforts on them. This ideal behaviour is not achieved in the BCGAs, since these algorithms handle all the digits in a similar way; often, the less significant digits converge towards an arbitrary value in the first stages. This causes the next search not to be efficient and thus the precision reached is not suitable.

One direction followed by BCGA research for achieving precision lies in executing the algorithm repeatedly, in such a way that the initial population of an execution ζ shall be made up by the *recoding* of the elements that belong to the final population generated in the execution $\zeta - 1$. The recoding is used for mapping these elements in a smaller search space, which is identified as a zone that is near the global optimum. Since during each run the action interval of the parameters is limited, the precision shall be more refined. Two techniques based on this idea for refining precision have been proposed: ARGOT (Shaefer, 1987), dynamic parameter encoding (Schraudolph et al., 1992) and delta coding (Whitley et al., 1991).

An effect that appears using the binary alphabet for representing elements in continuous domains is the so-called *Hamming cliff*. It is produced when the binary coding of two adjacent values differs in each one of their bits, for example, the strings 01111 and 10000 represent the values 31 and 32, respectively, and the values of each one of their

Figure 6. Fitness function with Hamming cliff



positions are different. The Hamming cliff may produce problems under some conditions, such as the convergence towards no global optimums. Figure 6 is an example of this (Goldberg, 1991a). As is shown in this figure, the elements placed in the left half have an above average fitness since $f(0 \star \star) > f(1 \star \star)$. A GA is fairly likely to converge initially to 011. The global optimum 100 is near to the element 011, however the Hamming distance between them is too high (three changes for accessing from 011 to 100 are required). This access is improbable when using the mutation operator (it is of $O(p_m^3)$). Therefore, convergence is likely to be produced towards the element 011. This problem may be solved by using the Gray code (Caruana et al., 1988), but doing so introduces higher order nonlinearities with respect to recombination, which causes the degree of implicit parallelism to be reduced (Goldberg, 1989b).

3.7.2. Redundance

When the binary alphabet is assumed for coding a parameter belonging to a finite discrete set with a cardinal different from a power of two, some codes may be *redundant*, e.g., their decodings correspond to values that don't belong to the domain of the parameter.

For example, consider a parameter $x_i \in S_i$ where S_i is a finite discrete set with $|S_i| = 10$. Using the binary alphabet, a minimum of four bits are needed for coding the elements in S_i . If codes 0000 to 1001 are selected for coding its elements, then, what do the codes 1010 to 1111 represent?

The existence of redundant codes states a problem since it may not be guaranteed that after the application of the crossover and mutation operators the codes generated are not redundant. Mechanisms to limit these types of codes in the population are needed. In (Beasley et al., 1993) the following possibilities are reviewed: 1) discard the redun-

dant codes generated or 2) assign low fitness to the chromosomes with redundant codes or 3) associate the redundant codes with valid ones. Solutions 1) and 2) may produce a loss of important genetic material simply because they link together redundant codes, which may induce poor GA behaviour. There are different mechanisms for carrying out the latter possibility, including *fixed remapping* and *random remapping*. With the former, each redundant code has a specific valid associated code to it. It is a very simple mechanism, but has the disadvantage that some values in the domain are represented by two different strings whereas the others only use one. With random remapping, each redundant code is associated with a valid code in a random way. This avoids the representational bias problem (Eshelman et al., 1989), but also causes less information to be passed on from parents to offspring. There is a hybrid alternative called *probabilistic remapping* in which every code (redundant or valid) is remapped to one of two valid codes in a probabilistic way, such that each valid code is equally likely to be represented.

3.7.3. Criticisms Against the Binary Coding Favours on the Implicit Parallelism Property

In the following we present three criticisms against the related advantage.

Schemata May Bear No Significance

Let S be a search space with $|S| = 2^L$ and B^L being the set of binary strings of length L . Then there are $2^L!$ possible representations of S using B^L associated with the set

$$\Theta = \{\theta | \theta : S \rightarrow B^L, \theta \text{ is injective}\}.$$

In (Liepins et al., 1990) theoretical results are presented that show the importance of selecting good representations (θ functions) and that these good representations are problem dependent. Knowing the best representation for a problem is tantamount to knowing how to solve the proper problem; in general the GA user cannot be expected to supply it. If a representation is randomly selected from Θ , perhaps the schemata shall bear no significance (Radcliffe, 1992) i.e., schemata don't relate chromosomes to correlated performance. Under these circumstances it should be clear that gathering information about the performance of any subset of the chromosomes (schemata) provides no information about the performance of the remaining ones. All schemata of the same order will tend to have roughly identical average fitness, and the evolutionary mechanism of the GA cannot distinguish among them (Tate et al., 1993) and so the search may not be effective.

Table V. Problem with deception

Chromosomes	Fitness
000	28
001	26
010	22
100	14
110	0
011	0
101	0
111	30

A particular case related with the non-significance of schemata is the so-called *deception* (Goldberg, 1989a). It follows from the schema theorem that the number of instances of a schema is expected to increase in the next generation if it is above-average and is not disrupted by crossover. Therefore, such schemata indicate the area within the search space that the GA explores and hence it is important that at some stage, these schemata contain the global optimum. Deception occurs when this is not true, since there are certain schemata that guide the search toward some solution that is not globally competitive. It is due since the schemata that have the global optimum don't bear significance and so they may not proliferate during the genetic process. Example 5 shows a problem where deception is present (Goldberg et al., 1989).

EXAMPLE 5. Let us consider the problem shown in Table V. For this problem the following relationships are hold:

$$\begin{array}{ll}
 f(0\star\star) > f(1\star\star) & f(00\star) > f(11\star), f(01\star), f(10\star) \\
 f(\star0\star) > f(\star1\star) & f(0\star0) > f(1\star1), f(0\star1), f(1\star0) \\
 f(\star\star0) > f(\star\star1) & f(\star00) > f(\star11), f(\star01), f(\star10)
 \end{array}$$

Although chromosome 111 is the global optimum we may predicte that a BCGA will have some difficulties in converging to it since almost none of the schema that contains this chromosome relates other chromosomes to correlated high performance. Moreover, there are many schemata that direct the search toward the chromosome 000 because they keep a good level of high performance correlation between their instances.

Inappropriate Schemata

A space with 2^L elements has 2^{2^L} subsets. If a representation based on an alphabet with k elements is used, there shall be a maximum of $(k+1)^n$ subsets that may be considered schemata. Subsets of the space may be built that group together promising chromosomes that share properties from which its good behaviour may be deduced. However, it is possible that these subsets don't form a schema, are not even contained in any schemata (except the most general one $\star\dots\star$) (Radcliffe, 1992). For example, coding the integers 0 to 15 using four digits, the representatives of 7 and 8 (0111 and 1000) share membership of no schema except $\star\dots\star$.

Not only schemata obey the schema Theorem

The maximum implicit parallelism degree reached by the binary alphabet is supported only if attention is restricted to classical schemata. This restriction is inappropriate since other schema definition under different coding types may be found allowing such a property to be also fulfilled (Antonisse, 1989; Radcliffe, 1991a; Vose, 1991).

4. Real-Coded Genetic Algorithms

In this section we treat the RCGA issues. We begin with the real coding study where different real coding models are presented. The advantages of the real representation are discussed. Then, mutation and crossover operators, that have appeared in the literature for working under real coding, are reviewed. An empirical comparison of some RCGAs built with different genetic operator configurations, is done. Mechanisms for tackling populations with elements belonging to constrained continuous spaces are described, and finally, some approaches for hybridization between RCGAs and other search methods are considered.

4.1. REAL CODING

As has already been pointed out, it would seem particularly natural to represent the genes directly as real numbers for optimization problems of parameters with variables in continuous domains. Then a chromosome is a vector of floating point numbers, the precision of which will be restricted to that of the computer with which the algorithm is carried out. The size of the chromosomes is kept the same as the length of the vector which is the solution to the problem; in this way, *each gene represents a variable of the problem*. The values of the chromosome genes

are forced to remain in the interval established by the variables which they represent, so the genetic operators must observe this requirement.

There are some representation models which are discarded from the previous one, *one gene-one variable*. In (Voigt, 1992) and (Voigt, 1993) a coding type is described under which each problem parameter has associated a number of m decision genes belonging to the interval $[0, 1]$. The chromosomes are formed by the link of the values of the decision genes in each parameter. For each parameter, the decoding process is carried out using a function $g : [0, 1]^m \rightarrow [0, 1]$, and a lineal transformation from the interval $[0, 1]$ to the corresponding parameter domain. As an example of such a function the authors presented the following:

$$\forall d = (d_1, \dots, d_m) \in [0, 1]^m \quad g(d) = \frac{1}{2^{m-1} - 1} \sum_{j=1}^m d_j 2^{j-1}.$$

When $m > 1$, this coding type breaks the one-to-one correspondence between genotype and phenotype, since two different genotypes may induce the same phenotype. So, it is impossible to find inferences from phenotype to genotype, i.e., the mapping from genotype to phenotype is not *isomorphic*. The idea followed by the authors is to emphasize the complexity of development from genotype to a mature phenotype occurring in nature. There are no one-gene, one-trait relationships in natural evolved systems. The phenotype varies as a complex, non-linear function of the interaction between underlying genetic structures and current environmental conditions. Nature follows the universal effects of *pleiotropy* and *polygeny*. Pleiotropy is the fact that a single gene may simultaneously affect several phenotype traits. Polygeny is the effect when a single phenotypic characteristic may be determined by the simultaneous interaction of many genes (Fogel, 1994). The model in (Voigt, 1992) and (Voigt, 1993) is an attempt to include this last effect in the GA's behaviour.

During the 1960s, a research line of algorithms based on natural evolution, called *Evolution Strategies* (ES) (Bäck et al., 1991a), was developed in Germany. The use of real representation is one of their main features. At first, mutation was the only recombination operator and the population was made up by only one element. Later, its study was extended by considering other recombination operators and populations with larger sizes. The chromosomes used by the ES are made up by a pair of float-valued vectors (x, σ) , where $x \in S$ (search space) and σ is a vector of standard deviations employed by the mutation operator. The mutation is executed by replacing x by $x + N(0, \sigma)$, $N(0, \sigma)$ being a vector of independent random Gaussian numbers with a mean of zero and standard deviations σ . One author (Mühlenbein et

al., 1993) proposed genetic operators for RCGA by extending existing operators for ES. In this paper, we study only the aspects related to the RCGAs, however we need to point out that the similarity between these algorithms facilitates the exchange of genetic operators between them.

4.2. REAL CODING ADVANTAGES

The use of real parameters makes it possible to use large domains (even unknown domains) for the variables, which is difficult to achieve in binary implementations where increasing the domain would mean sacrificing precision, assuming a fixed length for the chromosomes. Another advantage when using real parameters is their capacity to exploit the *graduality* of the functions with continuous variables, where the concept of graduality refers to the fact that slight changes in the variables correspond to slight changes in the function. In this line, a highlighted advantage of the RCGA is the capacity for the *local tuning* of the solutions. There are genetic operators such as *non-uniform mutation* (Michalewicz, 1992) that allows the tuning to be produced in a more suitable and faster way than in the BCGAs, where the tuning is difficult because of the Hamming cliff effect.

Using real coding the representation of the solutions is very close to the natural formulation of many problems, e.g., there are no differences between the *genotype* (coding) and the *phenotype* (search space). Therefore, the coding and decoding processes that are needed in the BCGAs are avoided, thus increasing the GA's speed. In (Radcliffe, 1992) it is suggested that a distinction between genotype and phenotype is not necessary for evolution. Thus, it is not justified that the definition of the genetic operators should be made upon the representation chosen. On the contrary, the author argued that whenever possible, *genetic operators and the analogues of schemata should be directly defined in the space for phenotypes*. The genetic operators and the schema concepts presented in the literature for RCGA agree with the Radcliffe's idea.

For Antonisse (Antonisse, 1989) binary coding is purposefully simple, while much of the work in the artificial intelligence community has been to develop highly expressive relatively complex representations. He presented a new schema interpretation that overturns the theoretical suitability of the binary alphabet in favour of the high cardinal alphabets. Further, for pointing out the importance of the expressiveness in the coding, he wrote:

"... This interpretation aligns theory with the intuition that the more expressive a language is the more powerful an apparatus for adapta-

tion it provides, and encourages exploration of alternative encoding schemes in GA research."

Clearly, since with the use of real coding the genotype and phenotype are similar, the expressiveness level reached is very high.

Along with these ideas, the relationship between the GAs and the *domain knowledge* need to be discussed. For Davis (Davis, 1989) most real-world problems may not be handled using binary representations and an operator set consisting only of binary crossover and binary mutation. The reason is that nearly every real-world domain has associated domain knowledge that is of use when one is considering a transformation of a solution in the domain. Davis believes that the real-world knowledge should be incorporated into the GA, by adding it to the decoding process or expanding the operator set. Real coding allows the domain knowledge to be easily integrated into the RCGA for the case of problems with non-trivial restrictions, as we shall see below.

4.3. CROSSOVER OPERATORS

Let us assume that $C_1 = (c_1^1 \dots c_n^1)$ and $C_2 = (c_1^2 \dots c_n^2)$ are two chromosomes that have been selected to apply the crossover operator to them. Below, the effects of different crossover operators for RCGAs are shown.

We should point out that since each crossover operator generate a different offspring number, a selection mechanism for deciding the ones that shall be included in the population is sometimes needed. This selection mechanism shall be called *offspring selection mechanism*.

Flat crossover (Radcliffe, 1991a)

An offspring, $H = (h_1, \dots, h_i, \dots, h_n)$, is generated, where h_i is a randomly (uniformly) chosen value of the interval $[c_i^1, c_i^2]$.

Simple crossover (Wright, 1991; Michalewicz, 1992)

A position $i \in \{1, 2, \dots, n - 1\}$ is randomly chosen and the two new chromosomes are built

$$H_1 = (c_1^1, c_2^1, \dots, c_i^1, c_{i+1}^2, \dots, c_n^2)$$

$$H_2 = (c_1^2, c_2^2, \dots, c_i^2, c_{i+1}^1, \dots, c_n^1)$$

Arithmetical crossover (Michalewicz, 1992)

Two offspring, $H_k = (h_1^k, \dots, h_i^k, \dots, h_n^k)$ $k = 1, 2$, are generated, where $h_i^1 = \lambda c_i^1 + (1 - \lambda)c_i^2$ and $h_i^2 = \lambda c_i^2 + (1 - \lambda)c_i^1$. λ is a constant (uniform arithmetical crossover) or varies with regard to the number of generations made (non-uniform arithmetical crossover).

BLX- α crossover (Eshelman et al., 1993)

An offspring is generated: $H = (h_1, \dots, h_i, \dots, h_n)$, where h_i is a randomly (uniformly) chosen number of the interval $[c_{min} - I \cdot \alpha, c_{max} + I \cdot \alpha]$, $c_{max} = \max(c_i^1, c_i^2)$, $c_{min} = \min(c_i^1, c_i^2)$, $I = c_{max} - c_{min}$. The BLX-0.0 crossover is equal to the flat crossover.

Linear crossover (Wright, 1991)

Three offspring, $H_k = (h_1^k, \dots, h_i^k, \dots, h_n^k)$ $k = 1, 2, 3$, are built, where $h_i^1 = \frac{1}{2}c_i^1 + \frac{1}{2}c_i^2$, $h_i^2 = \frac{3}{2}c_i^1 - \frac{1}{2}c_i^2$ and $h_i^3 = -\frac{1}{2}c_i^1 + \frac{3}{2}c_i^2$.

With this type of crossover an offspring selection mechanism is applied, which chooses the two most promising offspring of the three to substitute their parents in the population.

Discrete crossover (Mühlenbein et al., 1993)

h_i is a randomly (uniformly) chosen value from the set $\{c_i^1, c_i^2\}$

Extended line crossover (Mühlenbein et al., 1993)

$h_i = c_i^1 + \alpha(c_i^2 - c_i^1)$ and α is a randomly (uniformly) chosen value in the interval $[-0.25, 1.25]$.

Extended intermediate crossover (Mühlenbein et al., 1993)

$h_i = c_i^1 + \alpha_i(c_i^2 - c_i^1)$ and α_i is a randomly (uniformly) chosen value in the interval $[-0.25, 1.25]$. This operator is equal to the BLX-0.25.

Wright's heuristic crossover (Wright, 1990)

Let's suppose that C_1 is the parent with the best fitness. Then $h_i = r \cdot (c_i^1 - c_i^2) + c_i^1$ and r is a random number belonging to $[0, 1]$.

Linear BGA crossover (Schlierkamp-Voosen, 1994)

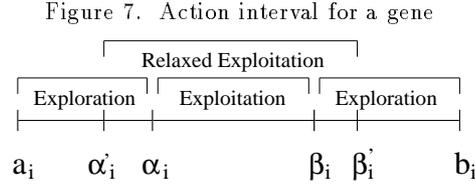
Under the same consideration as above, $h_i = c_i^1 \pm rang_i \cdot \gamma \cdot \Lambda$, where $\Lambda = \frac{c_i^2 - c_i^1}{\|C_1 - C_2\|}$.

The "-" sign is chosen with a probability of 0.9. Usually, $rang_i$ is $0.5 \cdot (b_i - a_i)$ and $\gamma = \sum_{k=0}^{15} \alpha_k 2^{-k}$ where $\alpha_i \in \{0, 1\}$ is randomly generated with $p(\alpha_i = 1) = \frac{1}{16}$. This operator is based on *Mühlenbein's mutation* (Mühlenbein et al., 1993).

Fuzzy Connectives Based Crossover (FCB) (Herrera et al., 1994)

In short, the interval of action of the gene i $[a_i, b_i]$ may be divided into three regions $[a_i, \alpha_i]$, $[\alpha_i, \beta_i]$, and $[\beta_i, b_i]$, where good descendents may be obtained; even considering a region $[\alpha'_i, \beta'_i]$ with $\alpha'_i \leq \alpha_i$ and $\beta'_i \geq \beta_i$ would seem reasonable. Figure 7 shows this graphically.

The intervals described above may be classified as *exploration* or *exploitation* zones. The interval with both genes being the extremes is an exploitation zone, the two intervals that remain on both sides are exploration zones and the region with extremes α'_i and β'_i could be considered as a *relaxed exploitation* zone.



We shall now go on to put forward a set of crossover operators that allow descendents to be obtained in the previous intervals. The variety of descendents in the interval of action guarantees the *population diversity*, and prevents the fast search space reduction, i.e., the *premature convergence*.

In order to do so, we shall use four functions F , S , M and L defined from $[a, b] \times [a, b]$ in $[a, b]$, $a, b \in \mathfrak{R}$, and which fulfill:

- (P1) $\forall c, c' \in [a, b] \quad F(c, c') \leq \min\{c, c'\}$,
- (P2) $\forall c, c' \in [a, b] \quad S(c, c') \geq \max\{c, c'\}$,
- (P3) $\forall c, c' \in [a, b] \quad \min\{c, c'\} \leq M(c, c') \leq \max\{c, c'\}$,
- (P4) $\forall c, c' \in [a, b] \quad F(c, c') \leq L(c, c') \leq S(c, c')$,
- (P5) $F, S, M, y L$ are monotone and non-decreasing.

Let us assume that $Q \in \{F, S, M, L\}$ and that $C_1 = (c_1^1 \dots c_n^1)$ and $C_2 = (c_1^2 \dots c_n^2)$ are two chromosomes that have been selected to apply the crossover operator to them. We may generate the chromosome $H = (h_1, \dots, h_i, \dots, h_n)$ as

$$h_i = Q(c_i^1, c_i^2), \quad i = 1, \dots, n.$$

T-norms, t-conorms, averaging functions and generalized compensation operators (Mizumoto, 1989a; Mizumoto, 1989b) may be used to obtain F , S , M and L operators. We shall associate F to a t-norm T , S to a t-conorm G , M with an averaging operator P , and L with a generalized compensation operator \hat{C} . In order to do so, we need a set of linear transformations to be able to apply these operators under the gene definition intervals.

Complying with a set of fuzzy connectives, (T, G, P, \hat{C}) , a set of associated functions F , S , M and L is built, which are described below:

If $c, c' \in [a, b]$ then

$$\begin{aligned}
F(c, c') &= a + (b - a) \cdot T(s, s') \\
S(c, c') &= a + (b - a) \cdot G(s, s') \\
M(c, c') &= a + (b - a) \cdot P(s, s') \\
L(c, c') &= a + (b - a) \cdot \hat{C}(s, s')
\end{aligned}$$

where $s = \frac{c-a}{b-a}$ y $s' = \frac{c'-a}{b-a}$.

These operators have the property of being continuous non-decreasing functions.

Making use of the previously proposed crossover operators, different RCGAs may be built, which are differentiated according to how they carry out the following two steps:

1. Generation of offspring using the different operators defined.
2. Selection of offspring resulting from the crossover which will form part of the population.

The proposal in (Herrera et al., 1994) was the following: For each pair of chromosomes from a total of $\frac{1}{2} * p_c * N$, four offspring are generated, the result of applying specific functions F , S , M , and L to them. All four offspring will form part of the population, in such a way that two of them substitute their parents and the other two substitute two chromosomes belonging to the remaining $\frac{1}{2}$ of the population that should undergo crossover. In this way, we include in the population two elements with exploration properties, one element with exploitation and another with relaxed exploitation. Thus, the exploration/exploitation relationship established shall bias the diversity.

We should point out that in (Karlin, 1968) a general crossover model for RCGAs was proposed called *nonrandom mating scheme*. The basic idea is to combine the two parents in a deterministic form; the offspring is a summation of its two parents scaled by two weighting matrices satisfying certain conditions of boundedness. The arithmetical and linear crossovers are two instances of these types of crossover models. In (Qi et al., 1994b) these models are generalized by allowing the weighting matrices to be formed by random values, i.e. a stochastic strategy. This new model includes the following crossover types:

1. the *discrete* one when the scaling matrices are diagonal containing only zeros and ones, and they add up to the identity matrix.
2. BLX- α and the two extended ones, when the weighting matrices are still diagonal but may contain elements other than zeros and ones and the two matrices may not add up to the identity matrix.

The effect of the crossover operators may be studied from two points of view: the gene level and the chromosome level. From the gene level, most crossover operators presented use the exploitation interval, which seems intuitively natural. However, studies made on BLX- α (Eshelman et al., 1993) and FCB crossover (Herrera et al., 1994) confirm the suitability of considering relaxed exploitation intervals as well. For the case of BLX- α in the absence of selection pressure all values $\alpha < 0.0$ will demonstrate a tendency for the population to converge towards values in the centre of their ranges, producing low diversity levels in the population and inducing a possible premature convergence toward non optimal solutions. Only when $\alpha = 0.5$ a balanced relationship between the convergence (*exploitation*) and divergence (*exploration*) is reached, since the probability that an offspring shall lie outside its parents becomes equal to the probability that it shall lie between its parents. Other types of crossover operators, besides BLX- α ($\alpha > 0.0$) and FCB crossover, handle genes that don't belong to the interval made up by the parent genes, such as the linear and the two extended crossovers.

For each crossover operator presented earlier, Table VI has a figure showing its effect on a gene i . Without any loss of generality, it is supposed that $c_i^1 \leq c_i^2$ and $f(C_1) \geq f(C_2)$.

From the chromosome level the effect of the crossover may be considered in a geometric way. Let us suppose two parents X and Y , we shall denote H_{xy} the hypercube defined by the component of X and Y . We may observe that the discrete and simple crossover generate a corner of H_{xy} , the extended intermediate crossover may generate any point within a slightly larger hypercube than H_{xy} , and the extended line and the linear crossover generate a point on the line defined by X and Y .

Finally, we consider the case of heuristic and linear BGA crossovers. Both include the goodness of the parents for generating the offspring. The heuristic one does it for determining suitable directions of the search process, it produces offspring in the exploration zone which is located at the side of the best parent. The linear BGA operator simulates the behaviour of Mühlenbein's mutation for generating offspring near to the best parent, it may generate them in exploitation or exploration zones, however the probability of visiting ones in the second type is very high.

4.4. MUTATION OPERATORS

Let us suppose $C = (c_1, \dots, c_i, \dots, c_n)$ a chromosome and $c_i \in [a_i, b_i]$ a gene to be mutated. Next, the gene, c_i' , resulting from the application of different mutation operators is shown.

Table VI. Crossover operators for RCGAs

Arithmetical	
BLX- α	
Linear	
Discrete	
Extended	
Wright's heuristic	
Linear BGA	
FCB	

Random mutation (Michalewicz, 1992)

c'_i is a random (uniform) number from the domain $[a_i, b_i]$.

Non-uniform mutation (Michalewicz, 1992)

If this operator is applied in a generation t , and g_{max} is the maximum number of generations then

$$c'_i = \begin{cases} c_i + \Delta(t, b_i - c_i) & \text{if } \tau = 0 \\ c_i - \Delta(t, c_i - a_i) & \text{if } \tau = 1 \end{cases}$$

with τ being a random number which may have a value of zero or one, and

$$\Delta(t, y) = y(1 - r^{(1 - \frac{t}{g_{max}})^b}),$$

where r is a random number from the interval $[0, 1]$ and b is a parameter chosen by the user, which determines the degree of dependency on the number of iterations. This function gives a value in the range $[0, y]$ such that the probability of returning a number close to zero increases as the algorithm advances. The size of the gene generation interval shall

be lower with the passing of generations. This property causes this operator to make a uniform search in the initial space when t is small, and very locally at a later stage, favouring local tuning.

Real Number Creep (Davis, 1991)

When a continuous function is optimized with local maximums and minimums and, at a given moment in time, a chromosome is obtained which is situated in a good local maximum, it would be interesting to generate other chromosomes around this chromosome in order to come close to the the peak point of such a maximum. In order to do this, we may slide the chromosome into a value which increases or decreases it by a small random quantity. The maximum slide allowed is determined by a parameter defined by the user. Different instances of this operator have been presented, such as the Guaranteed-Big-Creep and the Guaranteed-Little-Creep (Davis, 1989) and the small creep and the large creep (Kelly et al., 1991). The difference between these operators lies in the value of the maximum slide allowed.

Mühlenbein's mutation (Mühlenbein et al., 1993)

$$c'_i = c_i \pm rang_i \cdot \gamma,$$

where $rang_i$ defines the mutation range and it is normally set to $0.1 \cdot (b_i - a_i)$. The + or - sign is chosen with a probability of 0.5 and

$$\gamma = \sum_{k=0}^{15} \alpha_k 2^{-k},$$

$\alpha_i \in \{0, 1\}$ is randomly generated with $p(\alpha_i = 1) = \frac{1}{16}$.

Values in the interval $[c_i - rang_i, c_i + rang_i]$ are generated using this operator, with the probability of generating a neighbourhood of c_i being very high. The minimum possible proximity is produced with a precision of $rang_i \cdot 2^{-15}$.

The following operators are a generalization of the previous one. They only differ by how γ is computed.

Discrete modal mutation (Voigt et al., 1994)

$$\gamma = \sum_{k=0}^{\pi} \alpha_k B_m^k,$$

with $\pi = \lfloor \frac{\log(rang_{min})}{\log(B_m)} \rfloor$. $B_m > 1$ is a parameter called the base of the mutation and $rang_{min}$ is the lower limit of the relative mutation range.

Figure 8. Discrete modal mutation

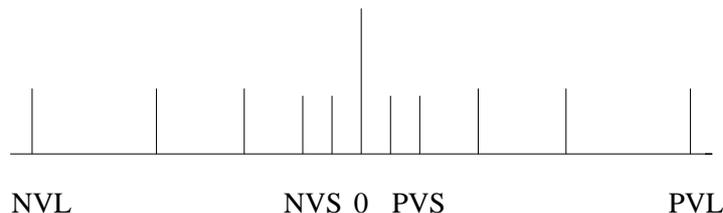


Figure 9. Continuous modal mutation

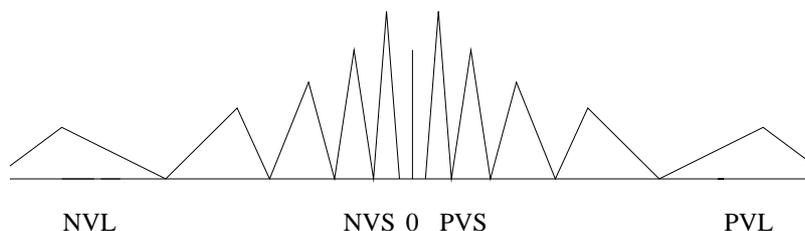


Figure 8 shows the possible change produced by this operator. NVS and PVS denote negative and positive very small mutations, respectively and NVL and PVL negative and positive very large mutations respectively.

Continuous modal mutation (Voigt et al., 1994)

$$\gamma = \sum_{k=0}^{\pi} \alpha_k \phi(B_m^k),$$

with $\phi(z_k)$ being a triangular probability distribution with $\frac{B_m^k - B_m^{k-1}}{2} \leq z_k \leq \frac{B_m^{k+1} - B_m^k}{2}$. Figure 9 shows the effects of this operator.

4.5. EXPERIMENTS

Minimization experiments on the functions in Table VII have been carried out. f_1 and f_2 were proposed in (De Jong, 1975) and f_3 in (Törn et al., 1989). We have considered $n = 25$ and the RCGAs shown in Table VIII. Table IX shows the connectives used by each one of FCB crossovers considered in Table VIII.

A binary-coded GA (BCGA) has also been included, which is based on a two point crossover. For this purpose, we used the GENESIS program (Grefenstette, 1990). The number of binary genes assigned to each variable is 11 for f_1 and f_2 and 31 for f_3 . Therefore, the precision, ρ , is approximately 10^{-6} .

Table VII. Test functions

Definition	Intervals	Optimum
$f_1(\vec{x}) = \sum_{i=1}^n x_i^2$	$-5.12 \leq x_i \leq 5.12$	$f_1^*(0, \dots, 0) = 0$
$f_2(\vec{x}) = \sum_{i=1}^{n-1} (100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	$-5.12 \leq x_i \leq 5.12$	$f_2^*(1, \dots, 1) = 0$
$f_3(\vec{x}) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-600.0 \leq x_i \leq 600.0$	$f_3^*(0, \dots, 0) = 0$

Table VIII. Real-Coded Genetic Algorithms

Algorithms	Mutation	Crossover
RCGA1	Random	Simple
RCGA2	Non-uniform ($b = 5$)	Simple
RCGA3	Random	Arithmetical ($\lambda = 0.5$)
RCGA4	Non-uniform ($b = 5$)	Arithmetical ($\lambda = 0.5$)
RCGA5- α	Non-uniform ($b = 5$)	BLX- α ($\alpha = 0, .15, .3, .5$)
RCGA6	Non-uniform ($b = 5$)	Linear
RCGA7	Mühlenbein	Discrete
RCGA8	Mühlenbein	Extended line
RCGA9	Mühlenbein	Extended intermediate
RCGA10	Modal Discrete ($B_m = 2, rang_{min} = 1.0e - 05$)	Extended intermediate
RCGA11	Modal Continuous ($B_m = 2, rang_{min} = 1.0e - 05$)	Extended intermediate
RCGA12	Non-uniform ($b = 5$)	Wright's heuristic
RCGA13	Mühlenbein	Linear BGA
RCGA14-Log	Non-Uniform	Logical FCB ($\psi = 0.5$)
RCGA14-Ham	Non-Uniform	Hamacher FCB ($\psi = 0.5$)
RCGA14-Alg	Non-Uniform	Algebraic FCB ($\psi = 0.5$)
RCGA14-Ein	Non-Uniform	Einstein FCB ($\psi = 0.5$)

We carried out our experiments using the following parameters: the population size is 61 individuals, the crossover probability $p_c = 0.6$, the probability of mutation $p_m = 0.005$, the parameter b used by the non-uniform mutation is 5, the selection procedure was linear ranking

Table IX. Fuzzy Connectives used by the FCB crossovers

FCB cros.	t-norm $T(x, y)$	t-conorm $G(x, y)$	Averaging Fun. $P(x, y)$ ($\psi \in [0, 1]$)	Gen. Comp. Op. $\hat{C}(x, y)$
Logical	$\min(x, y)$	$\max(x, y)$	$(1 - \psi)x + \psi y$	$T^{1-\psi} \cdot G^\psi$
Hamacher	$\frac{xy}{x+y-xy}$	$\frac{x+y-2xy}{1-xy}$	$\frac{1}{\frac{y-\psi y - xy + x\psi}{xy} + 1}$	$P(T, G)$
Algebraic	xy	$x + y - xy$	$x^{1-\psi} y^\psi$	$P(T, G)$
Einstein	$\frac{xy}{1+(1-x)(1-y)}$	$\frac{x+y}{1+xy}$	$\frac{2}{1+(\frac{2-x}{x})^{1-\psi}(\frac{2-y}{y})^\psi}$	$P(T, G)$

Table X. Results for f_1 (global optimum $f_1^* = 0$)

Algorithms	100	500	1000	5000	Online	Offline	Last Gen.
BCGA	9.1e+00	2.1e-01	2.2e+02	4.1e-05	1.3e+01	9.3e-01	4981
RCGA1	3.3e+00	1.1e-01	2.2e-02	8.2e-04	2.5e+00	5.4e-01	4826
RCGA2	5.8e+00	7.0e-02	1.1e-02	5.6e-15	1.1e+00	7.0e-01	4999
RCGA3	1.4e+00	8.2e-02	7.8e-03	1.3e-04	2.0e+00	1.8e-01	4468
RCGA4	1.4e+00	4.3e-02	3.4e-03	1.5e-13	4.5e-01	1.5e-01	4988
RCGA5-0.0	8.4e-01	2.1e-02	9.0e-04	1.7e-16	4.6e-01	1.4e-01	4999
RCGA5-0.15	5.0e-01	6.1e-03	2.2e-04	3.9e-18	4.9e-01	1.6e-01	4998
RCGA5-0.3	1.6e-01	2.9e-03	2.0e-04	1.5e-19	5.3e-01	1.7e-01	4999
RCGA5-0.5	6.2e-02	4.3e-07	2.4e-08	4.2e-23	6.5e-01	2.3e-01	4999
RCGA6	7.6e-01	3.0e-03	3.6e-04	1.4e-17	5.2e-01	1.3e-01	5000
RCGA7	3.4e+00	5.3e-04	3.6e-08	2.2e-09	6.0e-01	4.3e-01	1450
RCGA8	3.0e+00	7.3e-03	6.9e-06	2.0e-12	3.6e-01	2.4e-01	4842
RCGA9	5.1e-01	6.7e-05	3.3e-09	1.5e-13	2.9e-01	1.8e-01	4686
RCGA10	6.0e-02	5.4e-07	2.9e-08	4.1e-09	3.6e-01	1.5e-01	4776
RCGA11	3.1e-01	3.2e-05	1.8e-09	1.9e-13	3.1e-01	1.8e-01	4260
RCGA12	1.6e+01	3.2e-01	3.0e-02	1.4e-13	4.5e+00	1.3e+00	4999
RCGA13	4.8e+00	1.9e-02	1.9e-03	1.0e-05	3.2e+00	9.0e-01	4981
RCGA14-Log	1.8e+00	4.2e-02	1.8e-03	6.6e-17	5.6e-01	2.1e-01	4999
RCGA14-Ham	9.3e+00	8.5e-01	2.0e-01	4.1e-04	4.1e+01	7.2e-01	4975
RCGA14-Alg	1.9e+01	9.1e+00	4.6e+00	2.6e-02	9.1e+01	2.8e+00	4983
RCGA14-Ein	1.7e+01	1.3e+01	5.5e+00	4.4e-02	1.3e+02	3.0e+00	4976

(Baker, 1985) and elitist selection (De Jong, 1975) and the sampling model was stochastic universal sampling (Baker, 1987). We executed all the algorithms 5 times, each one with 5000 generations. For each function a table is presented below. From left to right the average values are shown for the best ones found in 100, 500, 1000 and 5000 generations, the *Online* measure (De Jong, 1975) (average of the fitness of all the elements appearing throughout the GA's execution), the *Offline* measure (De Jong, 1975) (average of the fitness of the best elements found in each generation) and finally the last generation in which the best element was found. The results are presented in Tables X, XI and XII.

Next, we analyse the results obtained.

4.5.1. Results Analysis

Looking back over the results, we may observe that BCGA shows no good behaviour. As it is expected, BCGA don't show an efficient progress on continuous search spaces.

The comparison between RCGA1 and RCGA2 on the one hand and RCGA3 and RCGA4 on the other allows the following conclusions to

Table XI. Results for f_2 (global optimum $f_2^* = 0$)

Algorithms	100	500	1000	5000	Online	Offline	Last Gen.
BCGA	2.1e+03	1.1e+02	3.6e+05	1.9e+01	1.3e+04	6.1e+02	4990
RCGA1	1.3e+03	1.1e+02	7.0e+01	5.5e+01	3.4e+03	4.3e+02	4917
RCGA2	1.3e+03	1.2e+02	7.7e+01	1.9e+01	1.0e+03	4.6e+02	4999
RCGA3	1.3e+02	4.1e+01	3.4e+01	2.3e+01	2.9e+03	1.2e+02	4868
RCGA4	1.8e+02	4.2e+01	2.3e+01	2.1e+01	4.8e+02	1.0e+02	4985
RCGA5-0.0	1.3e+02	3.7e+01	3.5e+01	3.1e+01	5.1e+02	1.2e+02	4999
RCGA5-0.15	1.8e+02	6.6e+01	5.3e+01	3.1e+01	5.6e+02	1.4e+02	4999
RCGA5-0.3	1.5e+02	6.1e+01	3.4e+01	2.0e+01	6.2e+02	1.6e+02	4999
RCGA5-0.5	9.9e+01	4.5e+01	4.2e+01	3.9e+01	8.1e+02	2.2e+02	4999
RCGA6	9.1e+01	2.5e+01	2.2e+01	2.0e+01	6.2e+02	1.0e+02	5000
RCGA7	1.3e+03	1.3e+02	3.8e+01	1.1e+01	5.9e+02	3.2e+02	4998
RCGA8	3.1e+02	8.9e+01	5.4e+01	3.1e+01	3.1e+02	1.4e+02	4999
RCGA9	1.8e+02	7.6e+01	4.9e+01	3.8e+01	3.3e+02	1.7e+02	5000
RCGA10	1.5e+02	4.5e+01	3.1e+01	2.9e+01	3.2e+02	1.5e+02	4999
RCGA11	1.9e+02	9.1e+01	6.6e+01	4.8e+01	3.7e+02	1.8e+02	4999
RCGA12	2.9e+03	1.4e+02	4.4e+01	1.5e+01	2.1e+03	7.6e+02	4999
RCGA13	2.8e+03	7.6e+01	2.8e+01	2.2e+01	3.4e+03	8.5e+02	4971
RCGA14-Log	3.8e+02	7.2e+01	5.5e+01	2.7e+01	5.8e+02	1.5e+02	4997
RCGA14-Ham	1.3e+03	1.4e+02	4.8e+01	4.0e-01	1.4e+04	2.1e+02	4986
RCGA14-Alg	3.7e+03	1.9e+03	8.0e+02	6.0e+01	7.9e+04	6.5e+02	4982
RCGA14-Ein	4.3e+03	1.3e+03	6.6e+02	7.0e+01	1.4e+05	5.8e+02	4985

be obtained between the random mutation (RM) and the non-uniform mutation (NUM).

1. With RM, good elements are reached in the initial stage of the GA (100 and sometime 500 generations). Since the whole definition interval is used for generating genes, the *exploration* level is high and very different zones are visited amongst them, with some having good elements.
2. NUM presents good results in subsequent stages (as from 1000 generations). With the passing of time, NUM reduces the genes generation interval in smaller zones around the gene to be mutated. The further the GA's execution advances the smaller the changes in the gene are. This produces a local tuning on the solutions, e.g., zones near to the best found solution are visited, which may be considered good enough for thinking that the optimal solution is close to them.
3. Worse results are obtained by RM in the final stages of the GA. Since the genes generation interval doesn't change throughout the GA's execution, during these stages, large changes in solutions close

Table XII. Results for f_3 (global optimum $f_3^* = 0$)

Algorithms	100	500	1000	5000	Online	Offline	Last Gen.
BCGA	3.3e+01	1.7e+00	7.5e+02	7.8e-02	4.5e+01	3.6e+00	4988
RCGA1	1.2e+01	1.4e+00	1.1e+00	3.7e-01	9.4e+00	2.6e+00	4821
RCGA2	2.1e+01	1.2e+00	1.0e+00	1.7e-02	4.3e+00	2.7e+00	4992
RCGA3	6.1e+00	1.3e+00	1.0e+00	1.1e-01	7.4e+00	1.1e+00	4471
RCGA4	5.3e+00	1.1e+00	9.4e-01	8.4e-03	1.9e+00	7.7e-01	4983
RCGA5-0.0	4.0e+00	1.0e+00	2.6e-01	7.4e-03	1.9e+00	6.9e-01	4983
RCGA5-0.15	2.9e+00	8.5e-01	1.5e-01	3.2e-02	2.0e+00	7.1e-01	4975
RCGA5-0.3	1.5e+00	7.8e-01	1.4e-01	4.7e-02	2.1e+00	7.4e-01	4955
RCGA5-0.5	1.2e+00	2.1e-02	2.0e-02	3.9e-03	2.4e+00	8.2e-01	3351
RCGA6	3.8e+00	9.3e-01	3.3e-01	3.4e-02	2.1e+00	6.5e-01	4991
RCGA7	1.3e+01	2.7e-01	3.9e-02	3.9e-02	2.3e+00	1.6e+00	1292
RCGA8	1.1e+01	8.7e-01	2.3e-02	2.0e-02	1.4e+00	9.4e-01	4674
RCGA9	2.8e+00	7.4e-02	2.5e-02	2.5e-02	1.2e+00	7.1e-01	4581
RCGA10	1.7e+00	4.8e-02	2.2e-02	2.2e-02	1.2e+00	6.7e-01	4329
RCGA11	2.0e+00	5.3e-02	1.4e-02	1.4e-02	1.2e+00	6.9e-01	4771
RCGA12	5.3e+01	1.2e+00	3.3e-01	2.1e-02	6.8e+00	4.2e+00	4982
RCGA13	1.7e+01	1.1e+00	8.5e-01	8.1e-02	1.2e+01	3.4e+00	4945
RCGA14-Log	7.3e+00	1.1e+00	7.7e-01	1.1e-02	2.3e+00	1.0e+00	4980
RCGA14-Ham	3.5e+01	3.4e+00	1.7e+00	2.3e-01	1.4e+02	3.2e+00	4984
RCGA14-Alg	7.3e+01	3.3e+01	1.1e+01	9.1e-01	3.1e+02	1.0e+01	4981
RCGA14-Ein	5.9e+01	3.6e+01	1.3e+01	1.0e+00	4.3e+02	9.7e+00	4983

to the optimum may be produced, in such a way that a loss of good solutions may occur.

In most functions, the algorithms based on Mühlenbein's mutation (RCGA7, RCGA8 and RCGA9) obtain good results in 100, 500 and 1000 generations, keeping low values for the *Online* and *Offline* measures as well. On the other hand, it may be observed how premature convergence is achieved in subsequent generations by seeing the last generations in which the best element was reached. With this mutation type, the probability of generating a gene close to the gene to be mutated is very high, in this way the local tuning may be processed at early GA's stages. However, the algorithms based on NUM generate high diversity levels during these stages, which supports the fact that later local tuning is carried out near to the optimum zones, producing a suitable convergence (see how with NUM, improvements appear until the final generation). Modal discrete and modal continuous mutation operators (RCGA10 and RCGA11 respectively) don't improve Mühlenbein's mutation (compare the results of RCGA10 and RCGA11 to RCGA9).

Generally, BLX- α crossover (RCGA5 family) allows the best final results to be obtained. It may be observed that the higher the α is, the better the results are. As α grows, the exploration level is higher, since the relaxed exploitation zones spread over exploration zones, increasing the diversity levels in the population. This allows good zones to be reached. Considering the final results for $\alpha = 0.5$ it seems natural that under this case an efficient *exploration and exploitation relationship* was induced. This relationship is not kept under the extreme case $\alpha = 0$, since the exploration property is not presented. If $\alpha = 0$, then there are no tendencies to visit new zones in the space, since the exploitation of the ones known is carried out. Therefore, the probability of premature convergence is very high.

Linear crossover (RCGA6) shows a good behaviour. The generation of one offspring in the exploitation zone and two in the exploration zone, along with the choice of the two most promising ones, guarantees the exploration and favours exploitation when generating good elements. In the first stages, the two explorative elements shall be used, thereby inducing diversity. In the final stages, the importance falls on the exploitative elements, under which the convergence is produced. We observe that the process of choosing the most promising elements *exploits* the knowledge about the zones visited, since it considers the fitness of the offspring for selecting the best. The principal drawbacks of this model with respect to the remaining ones is that it needs too many fitness function evaluations.

Simple and discrete crossover (RCGA2 and RCGA7) offer acceptable results as well. At the beginning, they biased diversity (observe the high final *Online* measure) since they exchange genes that are very different. When diversity disappears by means of the selection process, this interchange favours convergence.

The heuristic and linear BGA crossovers (RCGA12 and RCGA13 respectively) didn't show good behaviour. Perhaps this is due to the *overexploitation* of the search space, which induced a premature convergence towards good zones that are found in no final GA's stages. In this way, we should point out that in (Wright, 1990) it was advised the convenience of applying the heuristic crossover operators in advanced stages of the GAs.

Most algorithms in RCGA14 family returned low results; the replacement in the population of two parents by a set of three or four offspring produces too much diversity and thus slow convergence. To sum up, the exploration property related to this offspring selection mechanism leads to a high diversity level during the GA execution; the *Online* measure in this family is greater than in the remaining executed algorithms. The increase in the *Online* measure in RCGA14 family from the

RCGA14-Log algorithm to the RCGA14-Ein one suggests that diversity is greater using t-norm and t-conorm operators distant from the logical ones. This may be explained through the following order relation between the fuzzy connectives used in the definition of the FCB crossovers.

$$T_{Ein} \leq T_{Alg} \leq T_{Ham} \leq T_{Log} \leq G_{Log} \leq G_{Ham} \leq G_{Alg} \leq G_{Ein}$$

Among the FBC crossovers the logical crossover provides the lowest diversity levels. It is one of the best crossover operators compared. In this case, the combination of logical fuzzy connectives and the offspring selection mechanism considered offer a good exploration/exploitation relationship. The FCB crossovers may be used as tools to model the population diversity and so to avoid one of the most important problems with the GA: the *premature convergence*. Then, for a particular problem, it should be the user who can select the more suitable family of operators and also design new offspring selection mechanisms for establishing the best exploitation/exploration relationship for solving the problem. For example, the linear crossover may be considered as an instance of FCB crossover, where a t-norm, a t-conorm and an averaging function are used and the offspring selection mechanisms chooses the two most promising offspring.

4.5.2. *Final Remarks about Experiments*

Our final remarks about the previous experiments are the following:

1. Most RCGAs are best than BCGA.
2. Non-uniform mutation is very appropriate for RCGAs.
3. BLX- α (in particular $\alpha = 0.5$), logical FCB and linear crossovers have raised as the best crossover operators for RCGAs. An interesting property of all these crossover operators is that they consider the exploration intervals for obtaining offspring genes.

4.6. HANDLING CONVEX SPACES

The design of tools for handling non-trivial restrictions is easier using real coding. In (Michalewicz, 1992) an RCGA called GENOCOP is presented for solving optimization problems on continuous convex spaces, S_c , defined through a continuous space S along with an inequalities set R . The genetic operators used by GENOCOP assure that the chromosomes generated by them, considering those parents belonging to the space, belong to the space as well, i.e these operators are *closed*. This

property is not guaranteed in the BCGAs because of the restrictions introduced by R .

The operators used by GENOCOP are non-uniform, i.e., their action depends on the age of the population in which they are applied. Further, they are dynamic, i.e., the interval of possible values of a component i of a vector $X = (x_1, \dots, x_n) \in S_c$, $[a_i^x, b_i^x]$, is dynamic since the bounds a_i^x y b_i^x depend on the remaining values of the vector X and the set of inequalities R .

Using the properties fulfilled in a convex continuous space S_c

- i) $\forall x_1, x_2 \in S_c \forall \lambda \in [0, 1] \lambda x_1 + (1 - \lambda)x_2 \in S_c$ and
- ii) $\forall x \in S_c$ and any line l such that $x \in l$, l intersects the boundaries of S_c at precisely two points, a_l^x and $b_l^x \in S_c$,

the operators in GENOCOP are defined as follows:

Mutation Operators

Together with the random and non-uniform mutations, the following mutation operator is used:

- Boundary mutation

For a gene i in a chromosome C a value in $\{a_i^C, b_i^C\}$ is randomly chosen.

Crossover Operators

Together with the arithmetical crossover the following crossover operators are used:

- Modified simple crossover

For $C_1 = (c_1^1, c_2^1, \dots, c_n^1)$ and $C_2 = (c_1^2, c_2^2, \dots, c_n^2)$ two chromosomes to be crossed, a position $i \in \{1, \dots, n-1\}$ is randomly chosen and the two offspring are generated as follows:

$$H_1 = (c_1^1, \dots, c_i^1, \lambda c_{i+1}^2 + (1 - \lambda)c_{i+1}^1, \dots, \lambda c_n^2 + (1 - \lambda)c_n^1)$$

$$H_2 = (c_1^2, \dots, c_i^2, \lambda c_{i+1}^1 + (1 - \lambda)c_{i+1}^2, \dots, \lambda c_n^1 + (1 - \lambda)c_n^2)$$

where $\lambda \in [0, 1]$.

This operator is a generalization of the simple crossover for working with convex spaces.

– Single arithmetical crossover

Given a position $i \in \{1, \dots, n-1\}$ then

$$H_1 = (c_1^1, \dots, \omega c_i^1 + (1-\omega)c_i^2, \dots, \omega c_n^1 + (1-\omega)c_n^2)$$

$$H_2 = (c_1^2, \dots, \omega c_i^2 + (1-\omega)c_i^1, \dots, \omega c_n^2 + (1-\omega)c_n^1)$$

are generated, where ω is a random number belonging to the following range:

$$\omega \in \begin{cases} [\max(\alpha, \beta), \min(\gamma, \delta)] & \text{if } c_i^1 > c_i^2 \\ [0, 0] & \text{if } c_i^1 = c_i^2 \\ [\max(\gamma, \delta), \min(\alpha, \beta)] & \text{if } c_i^1 < c_i^2 \end{cases}$$

where

$$\alpha = \frac{b_i^{C_2} - c_i^2}{c_i^1 - c_i^2} \quad \beta = \frac{a_i^{C_1} - c_i^1}{c_i^2 - c_i^1}$$

$$\gamma = \frac{b_i^{C_1} - c_i^1}{c_i^2 - c_i^1} \quad \delta = \frac{a_i^{C_2} - c_i^2}{c_i^1 - c_i^2}$$

GENOCOP begins with an initial population made up by elements belonging to S_c and applies all the previous operators, with individual probabilities, having into account the dynamic ranges of the variables. Since these operators are closed, all elements appeared in subsequent populations shall be in S_c .

4.7. HYBRID REAL-CODED GENETIC ALGORITHMS

Different versions of hybrid RCGAs (HRCGAs) have been presented in the literature. Most approaches attempted to synthesize an RCGA along with a *hill climbing* method. One of these approaches is *BUGS* (Iba et al., 1992). In *BUGS*, each chromosome is a direction instead of a position, as happens in a conventional RCGA. The algorithm keeps a population of structures called *bugs*. Each *bug* is characterized by three elements:

$$bugs_i(t) = \begin{cases} \text{Position } x_i(t) = (x_1^i(t), \dots, x_n^i(t)) \\ \text{Direction } dx_i(t) = (dx_1^i(t), \dots, dx_n^i(t)) \\ \text{Energy } e_i(t) \end{cases}$$

where $x_i(t) \in S$, t is a given generation, and the energy $e_i(t)$ of the *bug* i is defined as

$$e_i(t) = \sum_{k=0}^{\tau_u} f(x_i(t-k)),$$

e.g., the cumulative sum of the fitness of the positions in the *bugs* i over the previous τ_u time steps generations, where τ_u is an input parameter.

In a generation t , the positions of the *bugs* are updated as follows:

$$x_i(t+1) = x_i(t) + dx_i(t).$$

Each *bug* may be considered to be a recording of the execution of a hill climbing procedure. The energy of a *bug* means the goodness of the results obtained by its associated hill climbing procedure in the last τ_u generations. The evolution of the *bugs* is driven by the values of their energies, in such a way that during the selection the only *bugs* that survive shall be the ones that offer the best results. The crossover and mutation operators act on the direction vectors of the *bugs*. The general process is the following:

P1. Generate an initial population at random

$$Pop(0) = \{bugs_1(0), \dots, bugs_N(0)\}.$$

Accept τ_u .

$t := 1$.

P2. For $i = 1, \dots, N$ Do $x_i(t+1) := x_i(t) + dx_i(t)$.

P3. For $i = 1, \dots, N$ Do $e_i(t) := \sum_{k=0}^{\tau_u} f(x_i(t-k))$.

P4. If t is a multiple of τ_u Then

P4.1. $n := 1$.

P4.2. Select two parents $bugs_i(t)$ and $bugs_j(t)$ using a probability distribution over the energies of all *bugs* in $Pop(t)$ so that *bugs* with higher energy are selected more frequently.

P4.3. With probability p_c apply the crossover operator to the $dx_i(t)$ and $dx_j(t)$ forming two offspring $bugs_n(t+1)$ and $bugs_{n+1}(t+1)$ in $Pop(t+1)$.

P4.4. Apply the mutation operator to $dx_i(t)$ and $dx_j(t)$ with probability p_m .

P4.5. $n := n + 2$.

- P4.6. If $n < N$ Then Go to P4.2.
- P5. If t is not a multiple of τ_u Then
- P5.1. $Pop(t + 1) := Pop(t)$.
- P5.2. Go to P2.
- P6. For $i = 1, \dots, N$ Do $e_i(t) = 0$.
- $t := t + 1$.
- Go to P2.

Another attempt was the *GA-Simplex* (Renders et al., 1994). The basic idea was to increase the local tuning of the RCGA by using a crossover operator called *simplex crossover*, which simulates the behaviour of a hill climbing method called *Simplex*.

5. Tools for the Analysis of the RCGA

Some authors attempted to generalize the schema theorem for the real coding case using previously a proper schema definition: Wright's Schemata (Wright, 1991), Formae (Radcliffe, 1991a; Radcliffe, 1991b) and the interval-schemata (Eshelman et al., 1993). Others presented mechanisms under which the behaviour of the RCGA may be explained or predicted, such as *virtual alphabet theory* (Goldberg, 1991a), the *expected progress* (Mühlenbein et al., 1993), and the study of RCGAs from a *stochastic process* point of view (Qi et al., 1994a; Qi et al., 1994b).

5.1. WRIGHT'S SCHEMATA

In (Wright, 1991) Wright studied the meaning of the classic binary-coded schemata for continuous functions in terms of real parameters. For a parameter i with domain $[a_i, b_i]$, a binary-coded schema ϵ of which all \star symbols are continuous at the right end of the string corresponds to a connected interval of real numbers $I_\epsilon \subseteq [a_i, b_i]$. These type of schemata are called *connected schemata*. Any simple parameter schema of which \star 's are not all continuous at the right end corresponds to a disconnected union of intervals. According to Wright, for most fitness functions, the *connected schemata* are the most meaningful in that they capture *locality information* about the function.

With this idea in mind, Wright defined a schema, ϕ , for handling real coding as

$$\phi = \prod_{i=1}^n (\alpha_i, \beta_i),$$

with $a_i \leq \alpha_i \leq \beta_i \leq b_i$. Under this definition Wright generalizes the schema theorem for the case of an RCGA, based on simple crossover and real number creep operators.

5.2. FORMAE

In (Radcliffe, 1991a) the notion of implicit parallelism is extended (and the associated schema theorem) to general non-string representations through the introduction of arbitrary *equivalence relations*. In doing so, it provides a framework within which arbitrary genetic operators may usefully be analyzed. Schemata are generalized to *formae* which are equivalence classes induced by equivalence relations over the search space. The aim is to maximize the predictive power of the schema theorem (and thus its ability to guide the search effectively) by allowing the developer of a GA for some particular problem to code knowledge about the search space by specifying families of formae (through the introduction of equivalence relations) that might reasonably be assumed to group together solutions with related performance and developing operators which manipulate these to good effect. Therefore, the author presents design principles for building useful equivalence relations, chromosome representations and crossover operators. Two of these principles are related to the crossover operator:

- *Respect*: Crossing two instances of any forma should produce another instance of that forma.
- *Proper assortment*: Given instances of two compatible formae (e.g., the ones where there are a chromosome being an instance of both), it should be possible to cross them to produce a child which is an instance of both formae.

In (Radcliffe, 1991b) four different types of formae in conjunction with operators for their effective manipulation are discussed. The first is related to the standard schemata, the second to the o-schemata introduced in (Goldberg, 1989a) to manage permutation problems, the other two are the *edge formae* defined to deal with chromosomes that represent edges (Whitley et al., 1989) and the *locality formae* which relate chromosomes on the basis of their closeness to each other and are applied to problems with continuous parameters.

In (Radcliffe, 1991a) the formae to manipulate real coding are discussed again. Two properties are considered to be captured by formae: locality and periodicity. The author presents two types of equivalence relations to induce formae describing these properties. The formae induced by the first type are like Wright's schemata. If these equivalence relations are to be used, then the crossover operator should be constructed which both respects and properly assort the formae they induce. Simple crossover (Wright, 1991; Michalewicz, 1992) would respect them, but fails properly to assort them. A more suitable crossover operator is the flat crossover which fulfills both principles. A GA using this might be expected to perform well on a real-valued problem for which locality is the appropriate kind of equivalence to impose on solutions, using the intrinsic parallelism which derives from each chromosome's being an instance of many locality formae. But a serious problem with this recombination operator is that, even with no fitness differences between any strings, the population will rapidly converge, at the centre of the range of each parameter. To help counteract this, Radcliffe also suggested *end point mutation* so as to reintroduce extreme values into the gene pool. Also, equivalence relations capturing periodicity are presented but no crossover operator exists that fulfills both of these principles. The author emphasizes that this is not a failure of the forma analysis, which has simply shown that general periodicities are extremely hard for a GA to be sensitive to.

5.3. INTERVAL-SCHEMATA

All computer solution methods require discretization. Given that computers have limited precision, any real-coded value may be mapped onto an integer. So in (Eshelman et al., 1993), it is suggested that the RCGAs may be more properly called integer-coded GAs (ICGAs).

For Eshelman et al. the real difference between a BCGA and an ICGA is that ICGA creates new individuals by operating on strings of integers rather than bit strings, since other differences such as the precision reached may be equalized. Obviously, every crossover operator presented for RCGAs is useful for ICGAs.

Analyzing the crossover operators for RCGAs it may be observed that they exploit the parameter intervals determined by the parents rather than the patterns of symbols they share. The classic binary-coded schemata were developed for strings of symbols, which is too restrictive for analyzing RCGAs. Something analogous is needed for RCGAs that manipulate intervals rather than bit values. From this idea the concept of *interval-schemata* arises. An *interval-schemata*, ι , is defined as:

$$\iota = \prod_{i=1}^n (\alpha_i, \beta_i),$$

with $\alpha_i, \beta_i \in \{0, \dots, 2^{L_i}\}$ and $\alpha_i \leq \beta_i$.

The similarity may be observed between this definition and Wright's schemata and the formae induced by the equivalence relations capturing locality.

Let us consider a parameter i with range $\{0, \dots, 2^{L_i}\}$. For this parameter there are $2^{L_i+1} - 1$ *connected schemata*, however there are $\frac{2^{L_i}(2^{L_i+1})}{2}$ *interval-schemata* which represent a greater quantity. This disadvantage implies that some intervals may not be represented by means of a *connected interval* such as [7, 10]. This problem doesn't appear when *interval-schemata* are used.

An important issue now is to study how the crossover operators preserve and explore *interval-schemata*. For a particular parameter, operators like the arithmetical and the BLX-0.0 crossover ones produce offspring that are members of the same interval-schemata in which the parents are common members (this is reconciled with Radcliffe's respect principle). However, these operators differ as to how many new interval-schemata are potentially reachable in a single crossover event. The arithmetical crossover is strongly biased towards certain interval-schemata over others. BLX-0.0, on the other hand, is much less biased in this respect, although it does have a bias towards points near the centre of the interval.

The BCGAs and the RCGAs are instances of two very general GA types: the so-called *symbol-processing* GAs (SPGAs) and *interval-processing* GAs (IPGAs), respectively. The way an IPGA processes interval-schemata is analogous to the way an SPGA processes symbol-schemata. For example, long interval-schemata correspond roughly to low order symbol-schemata. Both are characterized as not being very specific. As the search progresses, an SPGA shall progressively focus its search on higher order schemata whereas an IPGA shall progressively focus on shorter interval-schemata. In the former case, the SPGA has narrowed the search space down to certain partitions, whereas in the latter case the IPGA has narrowed the search to certain contiguous regions. As these values narrow, the search becomes more and more focused, taking its samples from smaller and smaller regions. So an IPGA exploits the *local continuities* of the function.

5.4. VIRTUAL ALPHABETS THEORY

The *virtual alphabet* theory (Goldberg, 1991a) postulates that because of selection, the available values of each gene are reduced in the initial generations, leaving only those belonging to some subsets of the initial domain associated with above-average fitness. Thus, virtually the RCGA uses low cardinality alphabets, the size of which is computed by the algorithm by means of *adaptation*.

5.4.1. Preliminary Definitions

Let us suppose $S = S_1 \times \dots \times S_n$ a search space, $f : S \rightarrow \Re$ an associated fitness function and $d(x)$, $x \in S$, a probability density function. Given an index set $I \subseteq \{1, \dots, n\}$, the marginal density function, $d_I(x_I)$, for the variables associated with I is calculated as:

$$d_I(x_I) = \int_{S_{I'}} d(x) dS_{I'},$$

where $I' = \{1, \dots, n\} - I$.

A *mean slice*, \bar{f}_I , or the expected value of f with respect to d and the free variables x_i ($i \in I$) is defined as

$$\bar{f}_I(x_I) = \int_{S_{I'}} d_{I'}(x_{I'}) f(x) dS_{I'},$$

where $dS_{I'} = \prod_{j \in I'} dx_j$ and $S_{I'} = \prod_{j \in I'} S_j$. If $|I| = 0$ then \bar{f}_I is reduced to \bar{f} , e.g., the expected value of f with respect to d . If $|I| = 1$, the slice shall be one-dimensional and shall be denoted as \bar{f}_j , with j being the only element of I . For example, $\bar{f}_1(x_1)$ is the expected value of f with respect to d for the single free variable x_1 .

Thus, essentially a set of functions, that are the averages of f with respect to any number of variables, is generated. In discrete GAs, this is the role played by schema fitness averages.

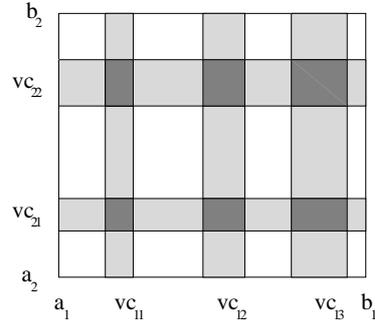
A *point slice*, $\bar{f}_I^p(x_I)$, with respect to the point $x = p$ is defined as the function obtained by setting $x_j = p_j$ $j \notin I$ and allowing the other variables to be varied freely. A *global slice*, $\bar{f}_I^*(x_I)$, is obtained when the point selected is the global optimum ($p = x^*$).

5.4.2. The Action of Selection

In (Goldberg et al., 1991b) it was shown that the number of generations, $\tau_{\bar{f}}$, required for above-average elements to dominate in the population may be calculated as

$$\tau_{\bar{f}} = \log_2 \log_2 N,$$

Figure 10. Effects of the selection and crossover



where N is the population size. For example for $N = 10^9$, $\tau_{\bar{f}}$ shall be approximately 5. The size of a typical population is from 30 to 1,000 elements, so $\tau_{\bar{f}}$ shall be equal to a value of three to four generations.

5.4.3. Virtual Characters and Alphabets

From the previous result, we may deduce that in first generations, the population tends to keep elements in S with higher fitness than \bar{f} . From then, each problem parameter i shall dispose of values, v_i , such as $\bar{f}_i(v_i) > \bar{f}$. These values form subsets of S_i called *virtual characters*, which build a *virtual alphabet*.

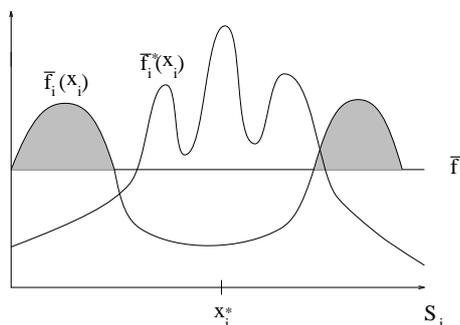
There is one consideration that we must take into account: the effects of selection along with crossover on virtual character processing. Using simple crossover, these effects are similar to the ones shown by GAs based on low cardinal alphabets. Figure 10 shows an example for a problem with two variables ($n = 2$).

We may observe that the intersection of the virtual characters of two variables delimits the zones that shall provide elements for the GA population. This situation could be changed by the effect of mutation. However, for mutation steps that are small with respect to the virtual character width, the errors made by assuming no change during the recombinative phase are small.

5.4.4. Blocking

Blocking is produced when the global optimum includes some parameter value, $x_i^* \in S_i$, that doesn't belong to any virtual character handled by the RCGA for S_i . In this way, there shall be inaccessible parameter values of the global optimum, since the search is centered on the parameter values which belong to existing virtual characters. Figure 11 shows an example of this problem.

Figure 11. Blocking



This figure shows \bar{f} , $\bar{f}_i(x_i)$, and $\bar{f}_i^*(x_i)$ for a particular parameter i in a problem with fitness function f . Each one of the dark zones is a virtual character. In the first generation, the search process shall limit the values of the i parameter, dealing only with the ones belonging to the existing virtual characters. Since x_i^* doesn't belong to any of these characters, it is ignored. Under these circumstances, it is said that the global optimum is blocked.

Blocking problems could be solved by using other mutation and crossover operators. The effect of these operators may be studied through the virtual alphabet theory. Goldberg studied two operators: random mutation and arithmetical crossover. Firstly, it may be thought that the former operator may solve the problem, because each point in the search space may be reached. However, this operator is very disruptive and can only be used with low probability. For a chromosome C to survive after mutation, the perturbations in C should produce a point at or above the current average fitness. There is reason to think that this shall not happen; the virtual characters are located where they are because the feature or features associated with that interval are of sufficient breadth and height to stick out above the crowd. It is very difficult to obtain an above-average point leading towards the optimum, not included in the virtual characters kept by the RCGA. In other words, the line search for random mutation to the global optimum is likely to fail because good features that are not close to already-represented virtual characters are like looking for a *needle in a haystack* (Goldberg, 1991a) with respect to that search. For the arithmetical crossover the conclusions were similar.

5.5. EXPECTED PROGRESS

In (Mühlenbein et al., 1993) a measure was presented called *expected progress*, in order to compare the behaviour of different mutation oper-

ator models for RCGA. Given an arbitrary point with distance ν to the optimum, the expected progress is defined as the expected improvement of x by successful mutation in Euclidean distance. This measure is defined by probability theory as an integral over the domain of successful mutations. The integrand is given by $progress(y) \cdot probability(y)$.

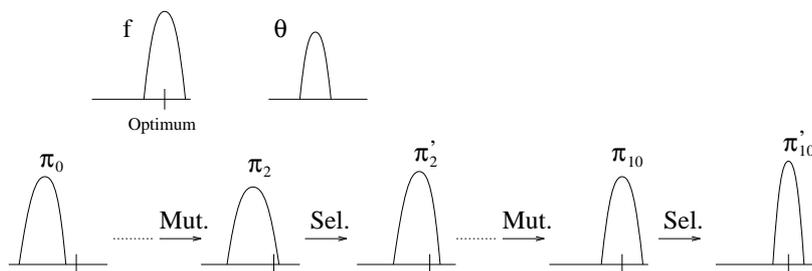
This approach is opposite to standard GA analysis, which starts off with the schema theorem, in which mutation and crossover are only considered as disruptions. Now these operators are evaluated according to the probability of generating better solutions.

The authors calculated the expected progress for three mutation operator types: Mühlenbein's mutation, random mutation and normal distributed mutation (similar to the one used in the *Evolution Strategies*). One of the main conclusions obtained was that the expected progress for Mühlenbein's mutation is 4-6 times lower than the expected progress for the normal distributed mutation with optimal σ , which is believed to be the optimal progress rate for search methods that do not use derivatives of the function (Mühlenbein et al., 1993).

5.6. RCGAS AS STOCHASTIC PROCESS

In (Qi et al., 1994a; Qi et al., 1994b) a general class of RCGA was analyzed, formulated as a discrete-time stochastic process (DTSP) of which states lay in a *multi-dimensional Euclidean* space. The study was carried out supposing *large population sizes*, this allows an easy mathematical treatment of the model to be made and important properties about the collective behaviour of the population as a whole to be obtained.

The use of a large population allows the authors to model the DTSP through a sequence of probability density functions, $\pi_k(\cdot)$ ($k \in N$), characterizing the distribution of the entire population, i.e., the frequency of occurrence of the elements in the search space appearing in the population. They derived time-recursive relationships for the population distribution. This is done by letting the population size N go to infinity and deriving the consequent limiting behaviour of the selection, crossover and mutation on the population distribution. Clearly, as the population size gets large, member points tend to cover the entire search space continuously; thus, the behaviour of the algorithm may be summarized by how dense the points are in the search space. First, Qi et al. studied the effect of the selection along with the mutation on $\pi_k(\cdot)$, then the ones for the crossover alone and finally the ones for all the operators together.

Figure 12. Effects of the selection and mutation on $\pi_k(\cdot)$ 

5.6.1. RCGA Model Studied

The properties of the RCGA class analyzed are:

1. Stochastic sampling with replacement selection.
2. Uniform crossover; after two parents are chosen, each corresponding pair of coordinates exchange their values independently, with the same probability, p_u . One of the two resulting vectors is arbitrarily discarded.
3. The mutation operator perturbs a chromosome following a common conditional probability density function $\theta_k(X|Y)$, symmetric in Y around X .

5.6.2. Selection and Mutation

In (Qi et al., 1994a) time-recursive formulae describing the behaviour of $\pi_k(\cdot)$ are derived, considering only the action of the selection and mutation operators. The results show how these operators play opposite roles: selection tends to squeeze $\pi_k(\cdot)$ around the global maximum of the fitness function, whereas mutation spreads the distribution. The joint effect of these operators may be observed in Figure 12, where mutation follows a Gaussian zero-mean distribution and the fitness function is unimodal.

The author pointed out the importance of selection through two results. The first one ensures under some continuity and connection around the global optimum conditions and supposing that $\pi_0(x^*) \neq 0$ (π_0 is the density function of the initial population and x^* the global optimum), the convergence of $\pi_k(x)$ towards $\delta(x - x^*)^3$ when $k \rightarrow \infty$, with $\delta(\cdot)$ being Dirac's function. The second one shows how selection alone increases the concentration of points in regions with current fitness above the population average:

LEMMA 1. Let B_k being the set of above-average vectors at generation k :

$$B_k = \{x \in S : f(x) \geq Esp_k\},$$

where $Esp_k = \int_S \pi_k(y)f(y)dy$ is the expected fitness at generation k , then the probability after selection of a member of the population to be in B_k is non-decreasing over time.

This result, which may be considered to be the continuous-space analog of schema theorem (Holland, 1975), shows how the selection process regards population members with above-average fitness.

For delimiting convergence conditions under the effects of the selection and mutation the authors presented a theorem establishing that *selection coupled with mutation can find the global optimum with a sequence of populations having increasing average fitness, as long as the mutation noise is small enough*. However, from a practical point of view (finite populations), we have to use a mutation noise as large as possible to guarantee sufficient coverage of the search space, in order to do so, we should find a sequence of mutation densities with the largest variance among those that guarantee convergence. Along this line, the authors derived sufficient conditions to obtain monotonically increasing average fitness, using only selection and mutation, for functions fulfilling the Lipschitz condition ($\forall x, y \in S |f(x) - f(y)| \leq L|x - y|$, $0 < L < \infty$). Basically, these conditions were fixed on the mutation densities. The study was restricted to mutation densities that are spherically symmetrical, in particular, conditions on a measure called *average radius* of the mutation ($\bar{r}(k, x) = \int_S \|y - x\|\theta_k(y - x)dy$) were obtained. It was interesting that these conditions were also expressed according to the statistics associated with the entire population. This suggests an *adaptive mutation scheme*, e.g., applying different mutation densities during the GA's generations, which shall be computed with information on each current population. This allows sufficient coverage and convergence to be reached throughout the GA's.

5.6.3. Crossover Alone

In (Qi et al., 1994b) the effects of the crossover operators isolated from the remaining operators were analyzed. Two theorems were introduced. The first predicts the effects of the application of these operators on $\pi_k(\cdot)$, whereas, the second postulates some asymptotic properties fulfilled after their application on a large number of generations. Next, we present this one.

THEOREM 3. *Repeated applications of the crossover operator lead to asymptotic independence among the coordinates with the marginal densities of each coordinate being unchanged.*

$$\lim_{k \rightarrow \infty} \pi_k(x_1, \dots, x_n) = \prod_{i=1}^n \pi_0(x_i),$$

where $\pi_0(x_i)$ represents the marginal density function of the component x_i .

Theorem 3 states that the crossover operator *exploits* the search space, breaking the correlations (*epistasis*) among its coordinates. The principal consequence of this effect is that all the cross central moments among the coordinates of the population vector shall converge to zero under repeated crossover.

Finally, the authors combined large sample results for selection and mutation with that for crossover in order to analyze the behaviour of the RCGA model. First, the selection and crossover union was studied, then, the three operators. The joint effects of selection and crossover provided a formal justification for the role of crossover during the global search: a particular region of the search space is sampled more often if any of the hyperplanes containing this region processes a higher average fitness. The intersection of the hyperplanes with this property allows good solutions to be achieved, from which more samples should be generated.

6. Conclusions

In this paper we have reviewed several issues relating to one of the most important alternatives to binary coding: *the real coding*. We have presented and compared the genetic operators described in the literature for GAs based on this type of coding. Tools that allow the behaviour of these algorithms to be studied have also been explained.

The most important feature of the RCGAs is their capacity to exploit local continuities, and the corresponding one of the BCGAs is their capacity to exploit the discrete similarities. Goldberg (Goldberg, 1991a) and Eshelman (Eshelman et al., 1993) leave to the user the decision for choosing one of these codings, suggesting that each one of them has suitable properties for different types of fitness functions. On the other hand, other authors such as Michalewicz (Janikow, et al., 1991; Michalewicz, 1992) defend the use of real coding, showing their advantages with respect to the efficiency and precision reached as compared to the binary one.

The experiments carried out highlight certain genetic operators as the most suitable ones for building RCGAs, such as the non-uniform mutation and the BLX- α , logical FCB and linear crossover operators. We should point out that these crossover operators may generate genes outside of the interval defined by the parent genes, which confirms the importance of considering the exploration and relaxed exploitation intervals for designing crossover operators for RCGAs.

Acknowledgements

We wish to thank the anonymous referees for their valuable comments which have improved the presentation of the paper.

References

- Antonisse, J. (1989). A new interpretation of schema notation that overturns the binary encoding constraint. *Proc. of the Third Int. Conf. on Genetic Algorithms*, J. David Schaffer (Ed.), (Morgan Kaufmann Publishers, San Mateo), 86-91.
- Bäck, T., Hoffmeister, F. & Schwefel, H-P. (1991a). A Survey of Evolution Strategies. *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, R. Belew and L.B. Booker (Eds.) (Morgan Kaufmann, San Mateo), 2-9.
- Bäck, T., Hoffmeister, F. & Schwefel, H-P. (1991b). Extended Selection Mechanisms in Genetic Algorithms. *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, R. Belew and L.B. Booker (Eds.) (Morgan Kaufmann, San Mateo), 92-99.
- Baker., J.E. (1985). Adaptive Selection Methods for Genetic Algorithms. *Proc. of an Int. Conf. on Genetic Algorithms*, (L. Erlbaum Associates, Hillsdale, MA), 101-111.
- Baker. J. E. (1987). Reducing bias and inefficiency in the selection algorithm. *Proc. Second Int. Conf. on Genetic Algorithms*, (L. Erlbaum Associates, Hillsdale, MA), 14-21.
- Beasley, D., Bull, D.R. & Martin, R. R. (1993). An Overview of Genetic Algorithms: Part 2, Research Topics. *University Computing*, **15**(4), 170-181.
- Belew, R.K. and Booker, L.B., (Eds.) (1991). Proceeding of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann Publishers.
- Booker, L.B., Goldberg, D.E. & Holland, J.H. (1989). Classifier Systems and Genetic Algorithms. *Artificial Intelligence* **40**(1/3), 235-282.
- Bramlette, M.F. (1991). Initialization, Mutation and Selection Methods in Genetic Algorithms for Function Optimization. *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, R. Belew and L.B. Booker (Eds.) (Morgan Kaufmann, San Mateo), 100-107.
- Bramlette, M.F. & Bouchard, E.E. (1991). Genetic Algorithms in Parametric Design of Aircraft. *Handbook of Genetic Algorithms*, L. Davis (Ed.) (Van Nostrand Reinhold, New York), 109-123.
- Caruana, R.A. & Schaffer, J.D. (1988). Representation and Hidden Bias: Gray versus Binary Coding for Genetic Algorithms. *Proc. of the Fifth International Conference on Machine Learning*, 153-162.
- Corcoran, A.L. & Sen S. (1994). Using Real-Valued Genetic Algorithms to Evolve Sets for Classification. *IEEE Conference on Evolutionary Computation*, 120-124.

- Cordon, O. & Herrera F. (1995). A General Study of Genetic Fuzzy Systems. *Genetic Algorithms in Engineering and Computer Science*, (Periaux, J., Winte, G., Eds.) John Wiley and Sons, 33-57.
- Davidor, Y. (1991). Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization. World Scientific, London.
- Davis, L. (1989). Adapting Operator Probabilities in Genetic Algorithms. *Proc. of the Third Int. Conf. on Genetic Algorithms*, J. David Schaffer (Ed.), (Morgan Kaufmann Publishers, San Mateo), 61-69.
- Davis, L. (1991). Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.
- De Jong, K.A. (1975). An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan.
- Eshelman, L.J., Caruana, A. & Schaffer, J.D. (1989). Biases in the Crossover Landscape. *Proc. of the Third Int. Conf. on Genetic Algorithms*, J. David Schaffer (Ed.), (Morgan Kaufmann Publishers, San Mateo), 86-91.
- Eshelman L.J. & Schaffer J.D. (1993). Real-Coded Genetic Algorithms and Interval-Schemata. *Foundation of Genetic Algorithms 2*, L.Darrell Whitley (Ed.), (Morgan Kaufmann Publishers, San Mateo), 187-202.
- Fogel, D.B. (1994). An Introduction to Simulated Evolutionary Optimization. *IEEE Trans. on Neural Networks* **5**(1), 3-14.
- Forrest, S. (Ed.) (1993). Proceeding of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, San Mateo.
- Forrest, S., Javornik, B., Smith, R.E. & Perlson, A.S. (1993). Using Genetic Algorithms to Explore Pattern Recognition in the Immune System. *Evolutionary Computation* **1**, 191-212.
- Fox, B.R. & McMahon, M.B. (1991). Genetic Operators for Sequencing Problems. *Foundations of Genetic Algorithms 1*, G.J.E. Rawlin(Ed.), (Morgan Kaufmann, San Mateo), 284-300.
- Goldberg, D.E. (1989a). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York.
- Goldberg D.E. (1989b). Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis. *Complex Systems* **3**, 153-171.
- Goldberg D.E., Korb, B. & Deb, K. (1989). Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems* **3**, 493-530.
- Goldberg D.E. (1991a). Real-Coded Genetic Algorithms, Virtual Alphabets, and Blocking. *Complex Systems* **5**, 139-167.
- Goldberg, D.E. & Deb, K. (1991b). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. *Foundations of Genetic Algorithms 1*, G.J.E. Rawlin(Ed.), (Morgan Kaufmann, San Mateo), 69-93.
- Goldberg, D.E. (1991c). Genetic and Evolutionary Algorithms Come of Age. *Communication of the Association for Computing Machinery* **37**(3), 113-119.
- Grefenstette J.J. (1990). A User's Guide to GENESIS Version 5.0.
- Grefenstette J.J. (Ed.) (1995). Genetic Algorithms for Machine Learning. Kluwer Academic Publishers, Boston. (Reprinted from *Machine Learning* **13**(2/3), 1993).
- Herrera, F, E. Herrera-Viedma., Lozano, M. & Verdegay, J.L. (1994). Fuzzy Tools to Improve Genetic Algorithms. *Proc. Second European Congress on Intelligent Techniques and Soft Computing*, 1532-1539.
- Herrera, F, Lozano, M. & Verdegay, J.L. (1995). Tuning Fuzzy Logic Controllers by Genetic Algorithms. *International Journal of Approximate Reasoning* **12**, 299-315.
- Holland, J.H. (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press.
- Holland, J.H., Holyoak, K.J., Nisbett, R.E. & Thagard, P.R. (1986). Induction. Processes of Inference, Learning, and Discovery. The MIT Press, Cambridge.

- Iba, H., Akiba, S., Higuchi, T. & Sato, T. (1992). BUGS: A Bug-Based Search Strategy using Genetic Algorithms. *Parallel Problem Solving from Nature 2*, R. Männer and B. Manderick (Eds.), (Elsevier Science Publishers, Amsterdam), 165-174.
- Ichikawa, Y. & Ishii, Y. (1993). Retainig Diversity of Genetic Algorithms for Multi-variable Optimization and Neural Network Learning. *Proc. IEEE Int. Conf. on Neural Networks*, San Francisco, California, 1110-1114.
- Janikow, C.Z. & Michalewicz, Z. (1991). An Experimental Comparison of Binary and Floating Point Representation in Genetic Algorithms. *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, R. Belew and L.B. Booker (Eds.) (Morgan Kaufmann, San Mateo), 31-36.
- Karlin, S. (1968). Equilibrium Behavior of Population Genetic Models with Non-random mating. *J. App. Prob.* **5**, 231-313.
- Kelly, J. & Davis, L. (1991). Hybridizing the GA and the K Nearest Neighbors Classification Algorithm. *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, R. Belew and L.B. Booker (Ed.) (Morgan Kaufmann, San Mateo), 377-383.
- Koza, J.R. (1992). Genetic Programing. The MIT press, Cambridge.
- Liepins, G.E. & Vose, M.D. (1990). Representational Issues in Genetic Optimization. *J. Expt. Theor. Artif. Intell.* **2**, 101-115.
- Liepins, G.E. & Vose, M.D. (1992). Characterizing Crossover in Genetic Algorithms. *Annals of Mathematics and Artificial Intelligence* **5**(1), 27-34.
- Lucasius, C. B. & Kateman G. (1989). Applications of genetic algorithms in chemometrics. *Proc. of the Third International Conference on Genetic Algorithms*, J. David Schaffer (Ed.), (Morgan Kaufmann Publishers, San Mateo), 170-176.
- Michalewicz, Z. (1991). A Genetic Algorithms for the Linear Transportation Problem. *IEEE Trans. on Systems, Man, and Cybernetics* **21**(2), 445-452.
- Michalewicz, Z. (1992). Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York.
- Michielssen, E., Ranjithan, S. & Mittra, R. (1992). Optimal Multilayer Filter Design Using Real Coded Genetic Algorithms. *IEE Proceedings-J*, **139**(6), 413-419.
- Mizumoto M. (1989a). Pictorial Representations of fuzzy connectives, Part I: Cases of t-norms, t-conorms and averaging operators. *Fuzzy Sets and Systems* **31**, 217-242.
- Mizumoto M. (1989b). Pictorial Representations of fuzzy connectives, Part II: Cases of Compensatory Operators and Seft-dual Operators. *Fuzzy Sets and Systems* **32**, 45-79.
- Mühlenbein H. & Schlierkamp-Voosen D. (1993). Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. *Evolutionary Computation* **1**, 25-49.
- Qi, X. & Palmieri, F. (1994a). Theoretical Analysis of Evolutionary Algorithms With an Infinite Population Size in Continuous Space Part I: Basic Properties of Selection and Mutation. *IEEE Trans. on Neural Networks* **5**(1), 102-119.
- Qi, X. & Palmieri, F. (1994b). Theoretical Analysis of Evolutionary Algorithms With an Infinite Population Size in Continuous Space Part II: Analysis of the Diversification Role of Crossover. *IEEE Trans. on Neural Networks* **5**(1), 120-128.
- Radcliffe N.J. (1991a). Equivalence Class Analysis of Genetic Algorithms. *Complex Systems* **5**(2), 183-205.
- Radcliffe N.J. (1991b). Forma Analysis and Random Respecful Recombination. *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, R. Belew and L.B. Booker (Eds.) (Morgan Kaufmann, San Mateo), 222-229.
- Radcliffe N.J. (1992). Non-Linear Genetic Representations. *Parallel Problem Solving from Nature 2*, R. Männer and B. Manderick (Ed.), (Elsevier Science Publishers, Amsterdam), 259-268.

- Reeves, C.R. (1993). Using Genetic Algorithms with Small Populations. *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, S. Forrest (Ed.), (Morgan Kaufmann, San Mateo), 92-99.
- Renders, J. M. & Bersini, H. (1994). Hybridizing Genetic Algorithms with Hill-Climbing Methods for Global Optimization: Two Possible Ways. *Proc. of The First IEEE Conference on Evolutionary Computation*, 312-317.
- Schlierkamp-Voosen D. (1994). Strategy Adaptation by Competition. *Proc. Second European Congress on Intelligent Techniques and Soft Computing*, 1270-1274.
- Schraudolph, N.N. & Belew, R.K. (1992). Dynamic Parameter Encoding for Genetic Algorithms. *Machine Learning* **9**, 9-21.
- Shaefer, C.G. (1987). The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique. *Proc. Second Int. Conf. on Genetic Algorithms*, (L. Erlbaum Associates, Hillsdale, MA).
- Syswerda, G. (1989). Uniform Crossover in Genetic Algorithms. *Proc. of the Third Int. Conf. on Genetic Algorithms*. Schaffer, J.D. (Ed.), (Morgan Kaufmann Publishers, San Mateo), 2-9.
- Syswerda, G. (1991). Schedule Optimization Using Genetic Algorithms. *Handbook of Genetic Algorithms*, L. Davis (Ed.) (Van Nostrand Reinhold, New York), 332-349.
- Tate, D.M. & Smith A.E. (1993). Expected Allele Coverage and the Role of Mutation in Genetic Algorithms. *Proceeding of the Fifth International Conference on Genetic Algorithms*, S. Forrest (Ed.), (Morgan Kaufmann Publishers, San Mateo), 31-36.
- Törn, A. & Antanas Ž. (1989). Global Optimization. Lecture Notes in Computer Science, Vol 350, Springer, Berlin.
- Voigt H. M. (1992). Fuzzy Evolutionary Algorithms. Technical Report tr-92-038, International Computer Science Institute (ICSI), Berkeley.
- Voigt H. M. (1993). A Multivalued Evolutionary Algorithm. Technical Report tr-93-022, International Computer Science Institute (ICSI), Berkeley.
- Voigt, H. M. & Anheyer, T. (1994). Modal Mutations in Evolutionary Algorithms. *Proc. of The First IEEE Conference on Evolutionary Computation*, 88-92.
- Vose, M. D. (1991). Generalizing the Notion of Schemata in Genetic Algorithms. *Artificial Intelligence* **5**, 385-396.
- Whitley, D., Starkweather, T. & Fuquay D. (1989). Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator. *Proc. of the Third Int. Conf. on Genetic Algorithms*, J. David Schaffer (Ed.), (Morgan Kaufmann Publishers, San Mateo), 133-140.
- Whitley, D., Mathias, K. & Fitzhorn, P. (1991). Delta Coding: An Iterative Search Strategy for Genetic Algorithms. *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, R. Belew and L.B. Booker (Ed.) (Morgan Kaufmann, San Mateo), 77-84.
- Whitley, L.D. & Schaffer, J.D. (1992). Proc. of the Int. Workshop on Combinations of Genetic Algorithms and Neural Network. IEEE Computer Society Press.
- Wright, A. (1990). Genetic Algorithms for Real Parameter Optimization. *Foundations of Genetic Algorithms, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems*, G.J.E. Rawlin (Ed.), (Morgan Kaufmann, Los Altos, CA), 205-218.
- Wright, A. (1991). Genetic Algorithms for Real Parameter Optimization. *Foundations of Genetic Algorithms 1*, G.J.E Rawlin (Ed.), (Morgan Kaufmann, San Mateo), 205-218.

Address for correspondence: Francisco Herrera, Dept. of Computer Science and A.I., ETS de Ingeniería Informática, University of Granada, 18071 Granada, Spain.