Soft Computing manuscript No.
(will be inserted by the editor)

Adaptive Representation for Single Objective Optimization

Crina Groşan, Mihai Oltean

Department of Computer Science
Faculty of Mathematics and Computer Science
Babes-Bolyai University
Kogalniceanu 1, Cluj-Napoca 3400, Romania

Received: date / Revised version: date

Abstract A new technique called Adaptive Representation Evolutionary Algorithm (AREA) is proposed in this paper. AREA involves dynamic alphabets for encoding solutions. The proposed adaptive representation is more compact than binary representation. Genetic operators are usually more aggressive when higher alphabets are used. Therefore the proposed encoding ensures an efficient exploration of the search space. This technique may be used for single and multiobjective optimization. We treat the case of single objective optimization problems in this paper. Despite its simplicity the AREA method is able to generate a population converging towards optimal solutions. Numerical experiments indicate that the AREA technique performs better than other single objective evolutionary algorithms on the considered test functions.

Keywords Evolution Strategy; Single Objective Optimization; Adaptive Representation; Higher Alphabets Encoding

1 Introduction

Adaptive Representation Evolutionary Algorithm (AREA) is similar to the Evolution Strategy (ES) technique ([11], [12]) as it uses a population of individuals which are modified by mutation. Whereas the ES individuals have a fixed representation (binary or real), the AREA individuals use an adaptive representation that may be changed during (and without halting) the search process.

ES employs a special mechanism for adapting the mutation parameter. For instance standard ES tries to adapt the standard deviation parameter when the Gaussian perturbation is used. These adaptations are of very little help. It is so because the function to be optimized is usually very intricate and the optimal parameter setting for a certain region of the search space may not be optimal for a neighboring region, at least.

Moreover, an incorrect setting for the value of the mutation parameter may lead to poor results. For instance, if the mutations are rare, the population could (and often will) converge to a local optimal point. If the mutations occur too often, the evolutionary process has a random search character. Facing these two problems, AREA employs a new way of searching through the solution space.

Many examples from nature can be found in order to sustain AREA. The first example resides in the human DNA which is, roughly speaking, a string of nucleotides over the alphabet {T (thymine), C (cytosine), G (guanine), A (adenine)} [6], [9]. By contrast, the standard evolutionary algorithms use strings over the alphabet {0, 1} which consists of only two values. The fournucleotide system has been developed under the specific conditions of the Earth environment. Had different conditions been on Earth, maybe a tennucleotide system could have developed. Lately, the entire evolution was based on this alphabet made up of only four symbols.

If we take a look at the history of the Earth we can see that the species evolved very slowly. Billions of years were needed to develop the diversity and perfection of life we know today. The entire evolution (of the complex structure) is based on reproduction by recombination and mutation. Recombination ensures the perpetuation of life. Mutation is responsible for maintaining diversity and for exploring new functional ways to combine the nucleotides.

If only twonucleotide systems had been used, the length of the human DNA (that encodes the same diversity) would probably have been very large. A mutation on this chromosome would probably be too small to produce a significant change. But, too many mutations would produce dramatic changes and the obtained individual would not survive.

If tennucleotide systems had been used, the length of the human DNA (that encodes the same diversity) would have (probably) been very small. A mutation on this chromosome would have produced a significant change and the species diversity would have been greater.

AREA is essentially a technique that works with higher alphabets. Each AREA individual consists of a pair (x, B) where x is a string encoding object variables and B specifies the alphabet used for encoding x. Binary encoded strings are a particular case of AREA.

If only one alphabet had been used the gain of AREA over standard ES would have been minimal. Thus, the AREA individuals use a dynamic system of alphabets that may be changed during (and without halting) the search process. If an individual gets stuck in a local optimum - from where it is not able to "jump"- , the individual representation is changed, hoping that this new representation will help the individual to escape from the current position and to explore farther and more efficiently the search space.

The similarities between the AREA behavior and the behavior of other species from nature are also numerous. For instance, the chameleon which is able to chance the color of its skin depending on the place. The AREA individuals possess the same ability to change their looks as the chameleon. From this point of view AREA may be considered as an interesting case of *chameleonic programming.*

Taking into account the No Free Lunch Theorems [16] (NFL) we cannot say that AREA is better than other evolutionary algorithms for all of the test problems. Indeed, several cases where other evolutionary algorithms used for comparison are better than AREA have been successfully identified. However, AREA significantly outperforms the standard evolutionary algorithms on the well-known difficult (multimodal) test functions. This advantage of AREA makes it very suitable for real-world applications where we have to deal with highly multi-modal functions.

Like GP and GA, AREA is also subject to a debate concerning the benefit of the genome-phenome systems over the genome only systems. As stated in many papers there is a question whether the maintaining of multiple, different genomes that encode the same phenotype should be beneficial. AREA maintains multiple different genomes (strings encoded over different alphabets) that encode the same phenotype (points in the search space) and this ability seems to be very beneficial.

AREA relies mainly on Dynamic Representation (DR) proposed in [10]. Both AREA and DR use higher alphabets for encoding solutions and a special mutation operator that changes the encoding alphabet during the search process. Whereas DR alphabets changing are blind, AREA employs an efficient strategy for changing the encoding alphabets.

The problems of adapting individual representation and the parameters of an evolutionary algorithm are difficult. They have been studied since the birth of genetic algorithms and evolutionary strategies. Some aspects of that study are described in [1] - [5], [13] - [15].

The paper is organized as follows. The AREA technique is described in section 2. The algorithms used in experiments are presented in section 3. Several numerical experiments using are performed in section 4. The test functions involved in the numerical experiments are well-known benchmarking problems used to asses the performances of the evolutionary algorithms. Most of these functions are highly multimodal employing different difficulties of the search space. Several important issues regarding the AREA representation are discussed in section 5.

2 The AREA technique

The main idea of this technique is to allow each solution be encoded over a different alphabet. Moreover, the representation of a particular solution is not fixed. Solution representation is adaptive and may be changed during the search process as an effect of the mutation operator.

2.1 Solution representation

Each AREA individual consists of a pair (x, B) where x is a string encoding object variables and B specifies the alphabet used for encoding x. B is an integer number, $B \geq 2$, and x is a string of symbols from the alphabet $\{0, 1, ..., B-1\}$. If $B = 2$, the standard binary encoding is obtained.

Each solution has its own encoding alphabet. The alphabet over which x is encoded may change during the search process.

When no ambiguity arises we will use B to denote the alphabet $B = \{0, 1, ...B-1\}$.

An example of an AREA chromosome is the following:

$$C = (301453,\ 6).$$

Remark: The genes of x may be separated by comma if required. For instance the comma separator is always needed when $B \geq 10$.

2.2 Mutation

Mutation can modify object variables as well as the last position (specifying the representation alphabet).

When the changing gene belongs to the object variable substring (x - part of the chromosome) the mutated gene is a randomly chosen symbol from the same alphabet.

Consider the chromosome C represented over the alphabet $B = 8$:

$$C = (631751,\ 8).$$

Consider a mutation occurs on position 3 in x and the mutated value of the gene is 4. Then the mutated chromosome is:

$$C_1 = (634751,\ 8).$$

If the position specifying B is changed, then the object variables will be represented by using symbols over the new alphabet, corresponding to the mutated value of B.

Consider the chromosome C represented over the alphabet $B = 8$:

$$C = (631751,\ 8).$$

Consider a mutation occurs on the last position and the mutated value is $B_2 = 10$. Then the mutated chromosome is:

$$C_2 = (209897,\ 10).$$

C and C_2 encode the same value over two different alphabets ($B = 8$, $B_2 = 10$).

Remarks:

(i) A mutation generating an offspring worse than its parent is called a *harmful mutation.*

(ii) A chromosome encoded over a higher alphabet has a shorter length than a chromosome encoding the same value (point in the search space) and the same precision but over a lower alphabet. For instance if we encode real numbers in the interval [0, 1] with the precision 10^{-9} we have to use strings of length 30 over the alphabet {0, 1} and strings of only 7 digits if we use the alphabet 30.

(iii) The alphabet part of a chromosome C it is not affected by normal mutation in our evolutionary model. The string x of the chromosome is the only one modified by normal mutation. The alphabet part of the chromosome is changed only when a predefined (consecutive) number of mutations of a solution do not improve the quality of the considered individual. However, one may consider mutations which affect the alphabet part of the individual in a standard way.

2.3 The evolutionary model

During the initialization stage each AREA individual (solution) is encoded over a randomly chosen alphabet. Each solution is then selected for mutation. If the offspring obtained by mutation is better than its parents, then the offspring enters the new population.

If the number of successive harmful mutations for an individual exceeds a prescribed threshold (denoted by MAX_HARMFUL_MUTATIONS), then the individual representation (alphabet part) is changed and it enters the new population with this new representation. Otherwise the individual (the parent) enters unchanged the next generation.

The reason behind this mechanism is to dynamically change the individual representation whenever necesary. If a particular representation has no potential for further exploring the search space, then the representation is changed. It is hoped that in this way the search space will be explored more efficiently.

2.4 AREA Parameters

The basic parameters of AREA are:

(i) The population size;

(ii) MAX_HARMFUL_MUTATIONS;

(iii) The alphabets over which an individual may be represented;
The sets of alphabets used in the experiments performed in this paper are {{0, 1}, {0, 1, 2}, ..., {0, 1, 2,..., 31}}.

(iv) The representation precision which is taken into account when the individual alphabet is changed into a new value;

(v) Mutation probability ($p_m$which is usually fixed (for instance 1 / mutations).

Note that if the alphabet is changed the p_m is changed. For instance, if the alphabet is $B = 2$ and the chromosome length is 30, the mutation probability is 1 / 30 = 0.033. If the used alphabet is 32 the chromosome encoding the same value is made of only 6 digits and $p_m = 1$ / 6 = 0.66.

2.5 The AREA algorithm

The AREA algorithm may be depicted as follows:

```
begin
Set $t = 0$;
Random initializes chromosome population $P$ (0);
Set to zero the number of harmful mutations for each individual in $P(0)$;
while ($t$ ¡ number of generations) do
begin
$P(t+1) = \emptyset$;
for $k = 1$ to PopSize do
begin
Mutate the $k^{th}$ (the current) individual from $P$ ($t$). An offspring is obtained.
Set to zero the number of harmful mutations for the offspring;
if the offspring is better than the current individual (the parent)
then the offspring is added to $P(t+1)$;
else begin
Increase the number of harmful mutations for the current individual;
```

```
    if the number of harmful mutations for the current individual = MAX_HARMFUL_MUTATIONS
    then begin
    Change the representation for the current individual;
    Set to zero the number of harmful mutations for the current individual;
    Add the current individual to $P(t+1)$;
    end
    else Add current individual (the parent) to $P(t+1)$;
    endif
    endif
    endfor;
    Set $t = t + 1$;
    endwhile;
    end
```

3 Algorithms used for comparison

Two algorithms are used for comparison purposes. Both of them use dynamic representation and have been adapted from [10]. Both algorithms use the evolutionary scheme as the one used by AREA (i.e. a population of individuals that are modified by mutation). Each of them is described below.

3.1 DRES algorithm

Dynamic Representation Evolution Strategy (DRES) uses the same solution representation and mutation operator as AREA uses. The difference between AREA and DRES consists in the technique for changing the base. In DRES algorithm the base is changed at the end of each generation with a fixed probability.

3.2 SMES algorithm

Seasonal Model Evolution Strategy (SMES) uses the same solution representation and mutation operator as AREA and DRES. In SMES algorithm the base in which solution is encoded is changed after a fixed (specified) number of generations.

4 Numerical Experiments

Several experiments are performed in this section by using 6 well-known benchmarking problems. In the first experiment we analyze the convergence ability to using different base for encoding solutions. For this purpose we use the bases 2 to 16. We apply the same algorithm (with the same basic parameters) for all considered bases of representation and for all considered test functions. The algorithm is an (1+1) ES and uses a population with a single individual in order to see the convergence speed.

In the second experiment the results obtained by AREA are compared with the results obtained by other two algorithms that use dynamic representation.

In third experiment we analyze how the number of alphabets used by AREA for encoding solutions influences the results.

Fourth experiment analyzes how the chose of the value for MAX_HARMFUL_MUTATIONS affects the results.

The essential role of these experiments is to show that using only one base for solution encoding (without change it during the search process) there are cases when the optimum cannot be found. Changing the representation base provides a new way of searching through the solution space. The second experiment show us which technique used for changing the base is suitable.

4.1 Test functions

The test functions used in these experiments are well known benchmarking problems used for assessing and comparing the performances of search algorithms [7], [17]. Six test functions are presented in this paper.

Each test function has one or more global optimal solutions and multiple local optimal solutions.

The test functions $f_1 - f_6$ are described in what follows. We denote by n the number of space variables, by $x = (x_i)_{i=1,n}$ a solution over the search space and by x^0 the global optimal solution.

Test function f_1 is the following:

$$f_1(x) = -a \cdot e^{-b\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}} - e^{\frac{\sum \cos(c \cdot x_i)}{n}} + a + e,$$

where $a = 20$, $b = 0.2$, $c = 2\pi$. The domain of definition is [-32, 32]^n.

The function is also known as Ackley's Path and is a widely used multimodal test function. The global minimum of this function is $x^0 = (0,0,\ldots,0)$ and the value of the function in this point is $f_1(x^0) = 0$.

Test function f_2 is defined as follows:

$$f_2(x) = \frac{1}{4000} \cdot \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1.$$

The domain of definition is [-500, 500]^n. Test function f_2, also known as Griewangk's function has many widespread local minima. The locations of the local minima are regularly distributed. The global minimum of this function is $x^0 = (0,0,\ldots,0)$ and the function's value in this point in $f_2(x^0) = 0$.

Test function f_3 is the following:

$$f_3(x) = -\sum_{i=1}^{n} \sin(x_i) \cdot \sin^{2\cdot m}(\frac{i \cdot x_i^2}{\pi}).$$

The domain of definition is $[0, \pi]^n$. The Michalewicz function (f_3) is a multimodal test function ($n!$ local optima). The parameter m defines the "steepness" of the valleys or edges. A higher value for m leads to more difficult search. For very large m the function is like a needle in the haystack (the function values for points in the space outside the narrow peaks give very little information on the location of the global optimum). The value of the function in the global optima (x^0) for n=10 is $f_3(x^0)$=-9.667

Test function f_4 is the following:

$$f_4(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)).$$

The domain of definition is $[-2,2]^n$.

Test function f_4 is also known as Rastrigin's function and is based on the unimodal function proposed by De Jong with the addition of cosine modulation to produce many local minima. Thus, the test function is highly multimodal. However, the locations of the minima are regularly distributed. The global optimum point is $x^0 =$ (0,0,...,0) and the value of the function in this point is $f_4(x^0) = 0$.

Test function $f_5$is the following:

$$f_5(x) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (1 - x_i)^2.$$

The domain of definition is $[-2, 2]^n$. Test function f_5, also known as Rosenbrock's valley, is a classic optimization problem, also known as the Banana function. The global optimum is inside a long, narrow, parabolically shaped flat valley. Finding the valley is trivial; still convergence to the global optimum is difficult. The global optimum point is x^0 = (0,0,...,0) and the value of the function in this point is $f_5(x^0) = 0$.

Test function f_6 is defined as follows:

$$f_6(x) = \sum_{i=1}^{n} (-x_i \cdot \sin(\sqrt{|x_i|})).$$

The domain of definition is $[-500, 500]^n$. Test function f_6 , also known as Schwefel's function is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to converge in the wrong direction. The global optimum point is x^0 =(420.9687,...,420.9687) and the value of the function in this point is $f_6(x^0)$ = $-n\cdot$ 418.9829.

The number of the space dimensions was set to 30 for each test function. Each algorithm is run 100 times for each test function in each experiment and with all of the considered parameters. Because individuals do not interact with one another, populations consisting of a single individual are used in all the experiments. The use of larger population sizes would bring about an increase in the performances. The representation precision was chosen in such a way as to have 30 digits for each space dimension when binary encoding is used. During the initialization stage (at the beginning of the search process) all of the AREA individuals are encoded over the alphabet {0, 1} (binary strings). One may initialize all of the AREA individuals over a randomly chosen alphabet.

The representation precision for the test functions f_1 - f_6 is presented in Table 1.

Table 1 The representation precision of the variables for the test functions f_1 - f_6.

Test function	Representation precision
f_1	0.0000001
f_2	0.000001
f_3	0.000000003
f_4	0.000000007
f_5	0.00000001
f_6	0.000001

4.2 Experiment 1

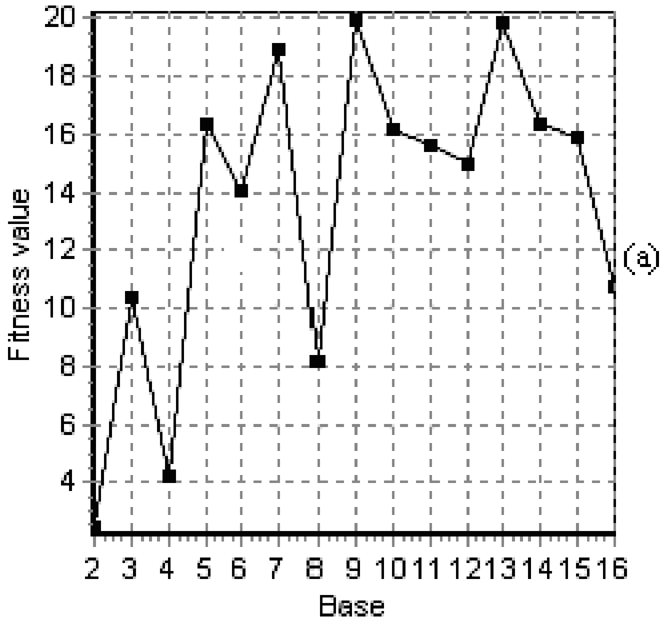
In this experiment the relationship between the base considered for solution representation and the average of the best individual value in the last generation is analyzed. The bases between 2 and 16 are used (more bases can be also used). Base 2 corresponds to binary encoding. Base 10 is similar to real encoding, but the corresponding genetic operators are different from those used in conjunction with the real encoding. The algorithms are run for 10000 iterations. The number of iterations is not sufficient to ensure a perfect convergence but it can lead us to the base suitable for each function. The results are averaged over 100 runs.

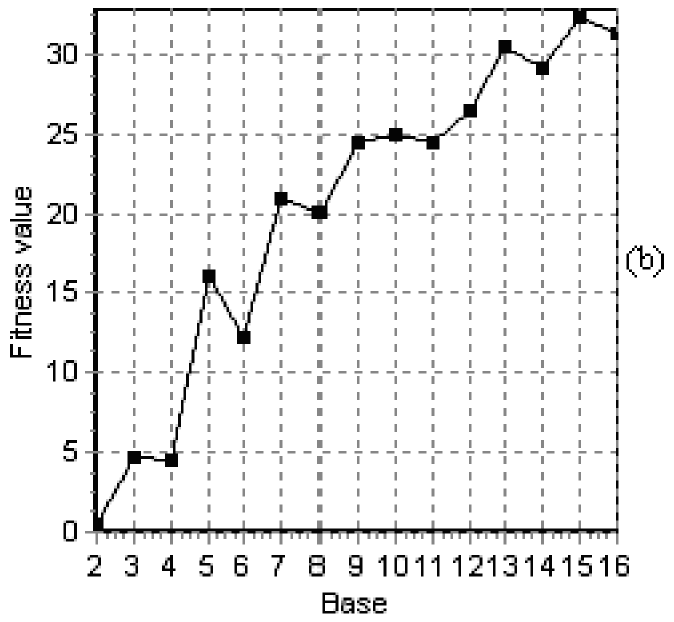
The algorithm used is a simplified (1+1) ES. which uses a single population of individuals that are represented as binary strings. In our experiments we allow solutions be encoded in other bases (not only in base 2). The genetic operator used in this algorithm is the mutation only. Each individual is selected for mutation. Each gene is mutated with a fixed mutation probability. The parent and the offspring compete for survival.

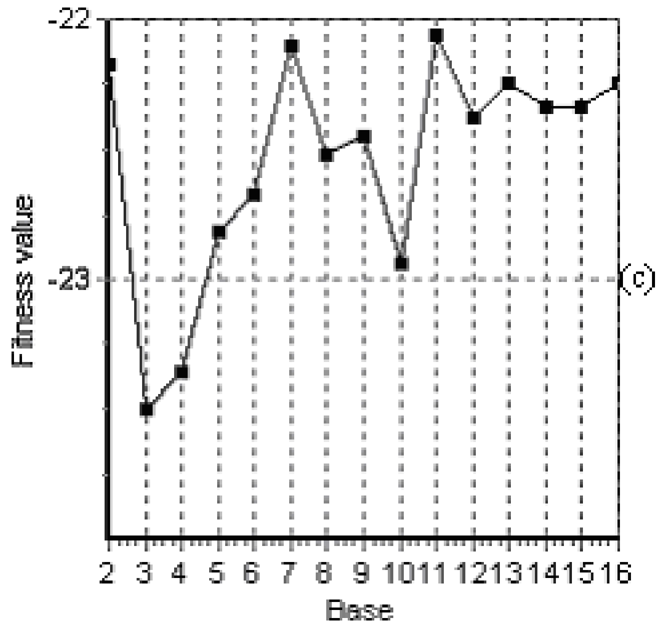
The parameters values used in this algorithm are given in Table 2.

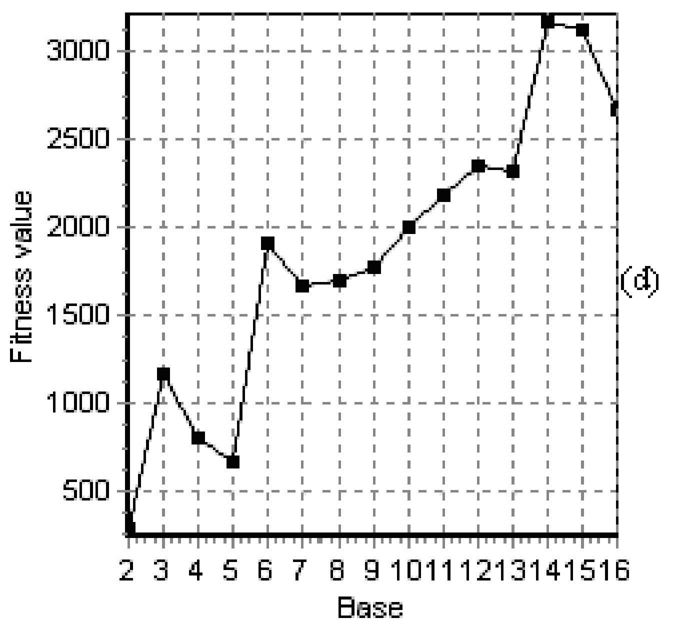
The relationship between fitness value and the base used for encoding the chromosomes is depicted in Figure 1.

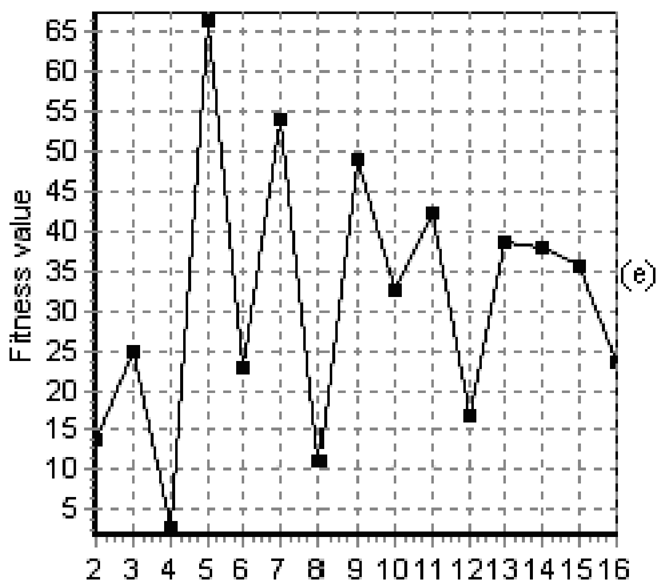
Figure 1 (a) shown that after 10000 iterations binary encoding seems to be better than the encoding over the other considered bases. Thus we can choose binary encoding for solving this problem. The results obtained by considering base 4 is close to the result obtained using binary encoding.


Fitness value
20
18
16
14
12
10
8
6
4
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Base
(a)


Fitness value
30
25
20
15
10
5
0
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Base
(b)


Fitness value
-22
-23
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Base
(c)


Fitness value
3000
2500
2000
1500
1000
500
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Base
(d)


Fitness value
65
60
55
50
45
40
35
30
25
20
15
10
5
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
(e)

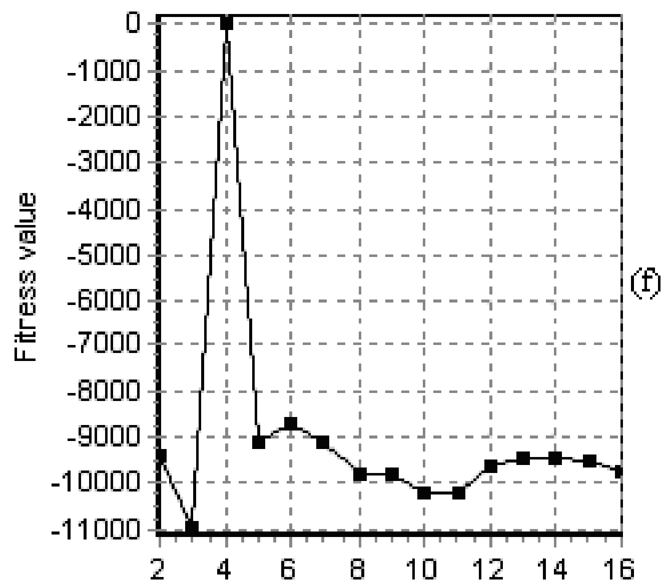

Fitness value
0
-1000
-2000
-3000
-4000
-5000
-6000
-7000
-8000
-9000
-10000
-11000
2 4 6 8 10 12 14 16
(f)

Table 2 Parameter used by (1+1) ES for the test function $f_1 - f_6$.

Parameter	**Value**
Number of dimensions	30
Number of mutations per chromosome	1
Population size	1
Number of generations	10000

According to Figure 1 (b) binary encoding seems to be better than the encoding over the other considered bases. Thus we can choose binary encoding for solving this problem. The results obtained by using the bases 3 and 4 are also good results.

As we can see from Figure 1 (c) the best fitness value for 10000 generations is obtained by considering solutions encoded in the bases 3 and 4. We can see that almost considered base representations give e better result than binary representation (without bases 7 and 11).

Figure 1 (d) shown that the algorithm using binary encoding performs significantly better than the algorithm using other encoding. Good results are obtained considering bases 4 and 5.

We can see from Figure 1 (e) that the best result is obtained by encoding solutions in the base 4. A good result is obtained also for the base 8 and 2.

For the test function f_6 (see Figure 1 (f) the base 3 seems to be the best choice for encoding solutions. The results obtained considered the bases 8, 9 10 11 and 12 are better than binary encoding.

4.3 Experiment 2

The relationship between the number of iterations and the average of the best individual value in the last generation is analyzed in this experiment. The parameters used by AREA are given in Table 3.

Table 3 Parameters used by AREA for Experiment 2.

Parameter	Value
Number of alphabets	31
MAX_HARMFUL_MUTATION	50
Number of mutations / chromosome	2

DRES and SMES use the same parameters as AREA. The probability of changing an alphabet in DRES is 0.02. The number of generations after the passing of which SMES changes the alphabet is 50.

The results of this experiment are depicted in Figure 2.

To obtain a stronger evidence of the AREA power we give in Table 4 the mean of the results at the end of search.

Table 4 The average (over 100 runs) of the solutions at the end of search.

Test function	AREA	DRES	SMES
f_1	0.9524	2.7255	2.6234
f_2	0.2299	0.7221	0.7177
f_3	-26.4601	-26.8517	-26.5056
f_4	8.1332	9.1527	9.8337
f_5	137.0539	164.3271	155.4578
f_6	-11720.6633	-11962.1233	-11956.92

From Figure 2 and Table 4 we can see that AREA outperforms DRES and SMES algorithms on most of the considered test problems. DRES and SMES have a better overall behavior than AREA only for the test function f_6. AREA also has a faster convergence than DRES and SMES.

To determine if the differences between AREA and DRES or SMES are significant we use a t-test with 95% confidence. Before applying the t-test an F-test has been used for determining whether the compared data have the same variance. The P-values of a two-tailed t-test are given in Table 5.

Table 5 The P-values of the t-test with 99 degrees of freedom.

Test function	AREA – SMES P – Values	AREA – DRES P - Values
f_1	1.7E-11	1.1E-10
f_2	1.6E-6	1E-7
f_3	0.8000	0.0396
f_4	0.0446	0.2098
f_5	0.4081	0.2078
f_6	0.0122	0.0142

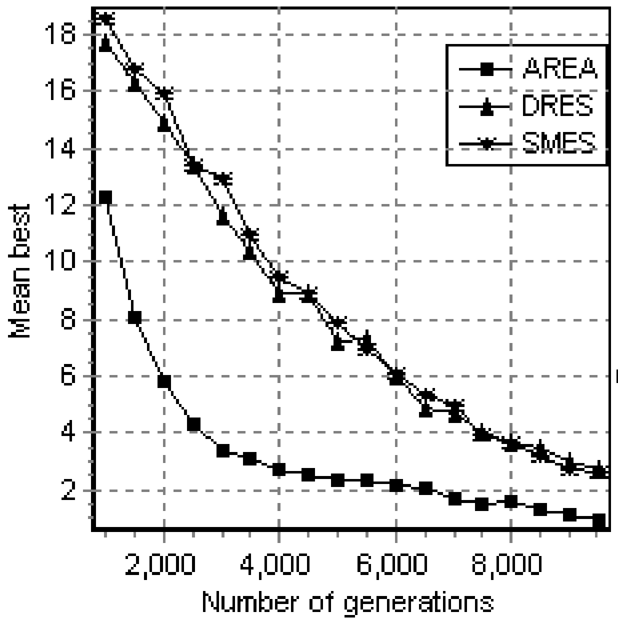
From Table 5 it can be seen that the differences between AREA and the other two algorithms are statistically significant on most of the considered test functions excepting the test functions f_3 and f_5 (where the differences between AREA and SMES are not considered statistically significant) and for the functions f_4 and f_5 (where the differences between AREA and DRES are not considered statistically significant).

4.4 Experiment 3

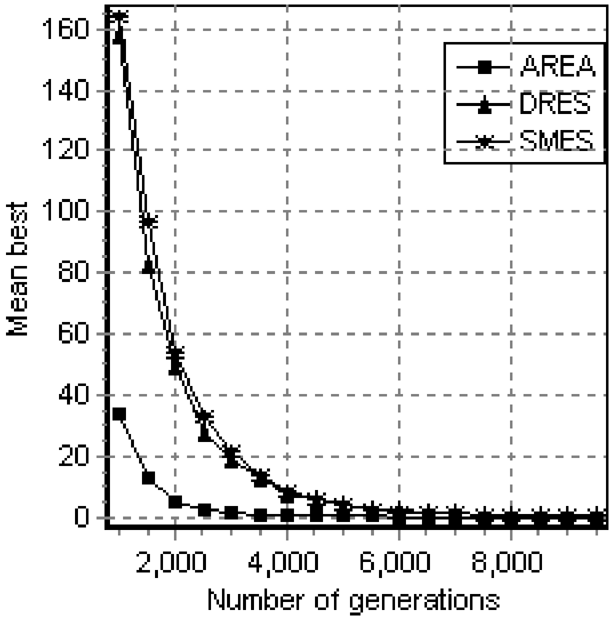
The relationship between the number of alphabets used for chromosome encoding and the average of the best individual values in the last population is analyzed in this experiment. The parameters used by AREA are given in Table 6.

For speed purposes, the test functions are analyzed for 10 dimensions.

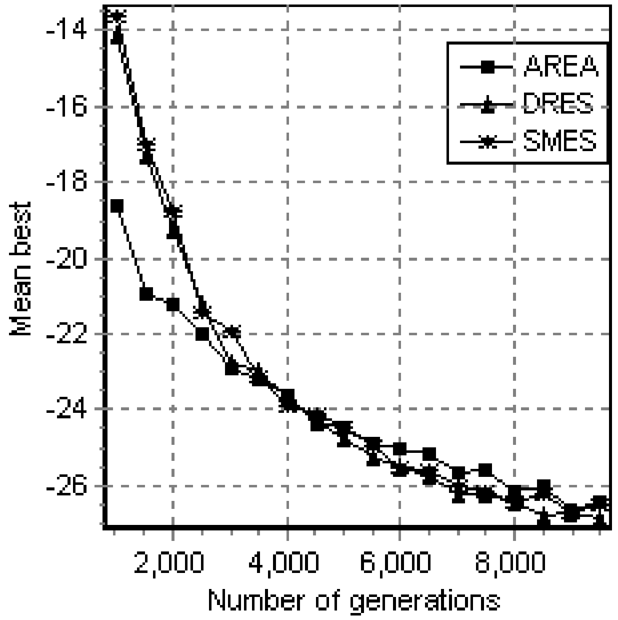
In Figure 3 the results of this experiment are depicted.


AREA
DRES
SMES
Mean best
Number of generations

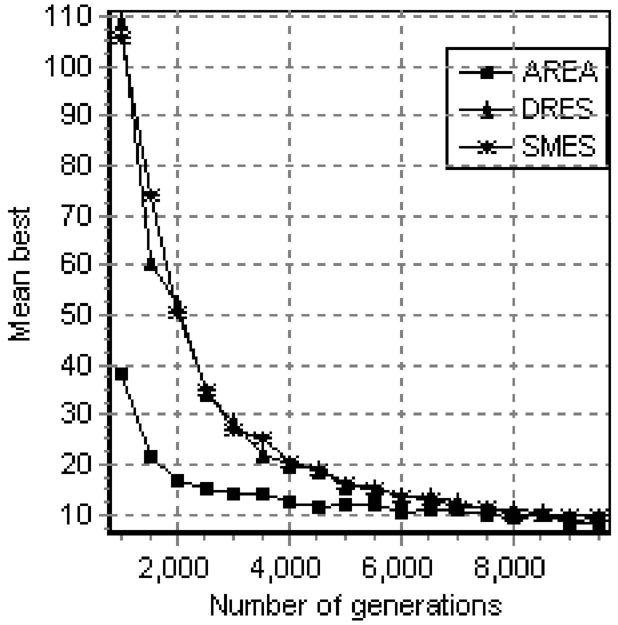
(a)


AREA
DRES
SMES
Mean best
Number of generations

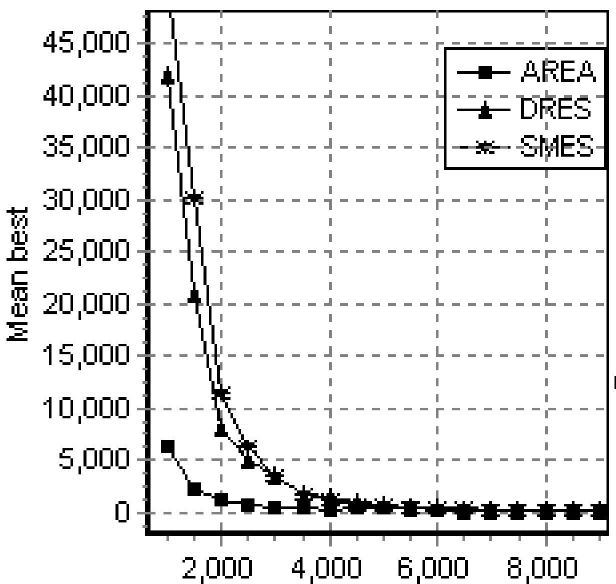
(b)


AREA
DRES
SMES
Mean best
Number of generations

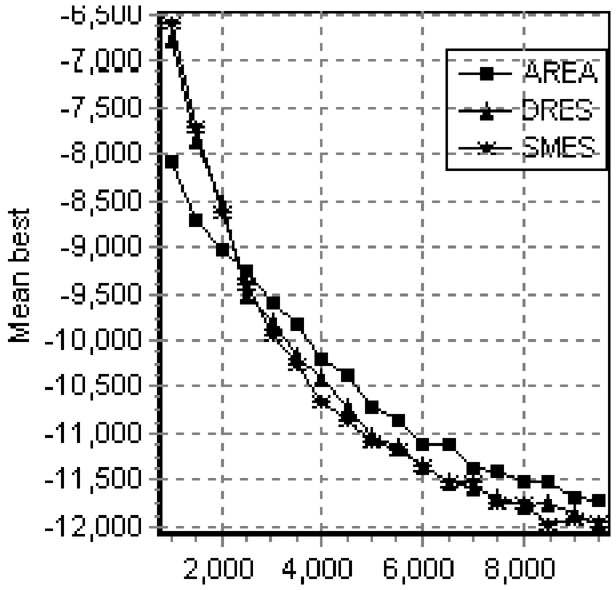
(c)


AREA
DRES
SMES
Mean best
Number of generations

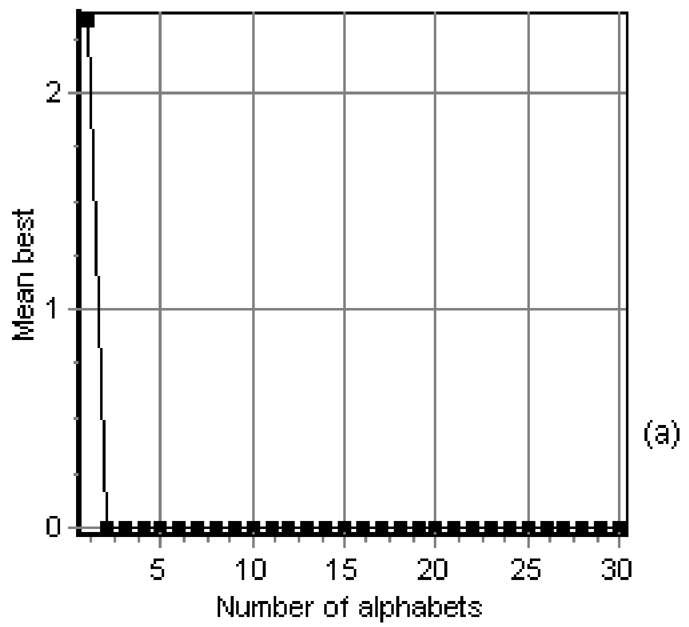
(d)

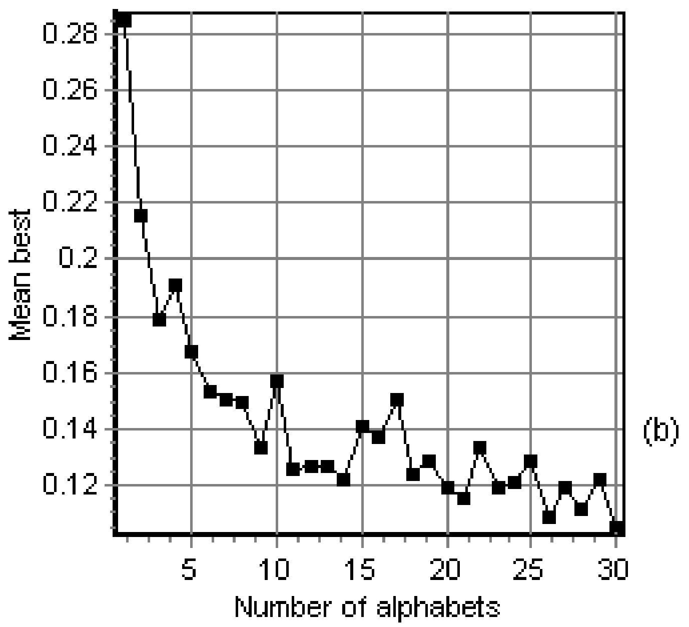

AREA
DRES
SMES
Mean best

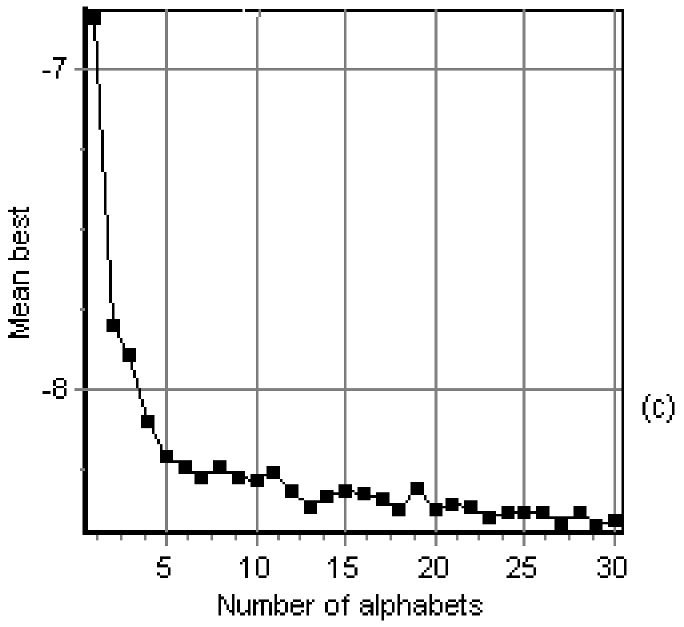
(e)

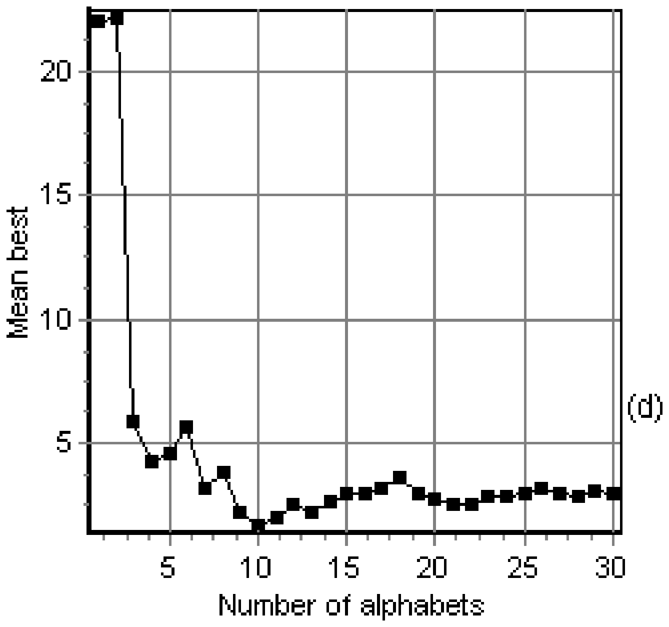

AREA
DRES
SMES
Mean best

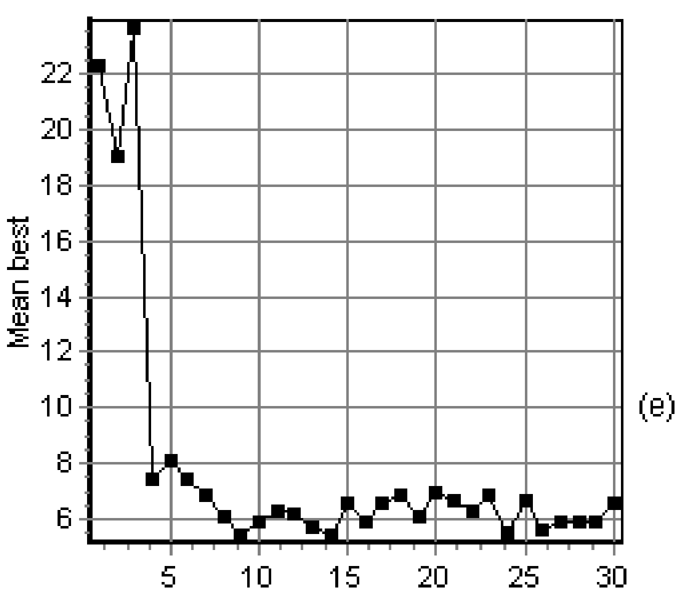
(f)


Mean best
Number of alphabets
(a)


Mean best
Number of alphabets
(b)


Mean best
Number of alphabets
(c)


Mean best
Number of alphabets
(d)


Mean best
(e)

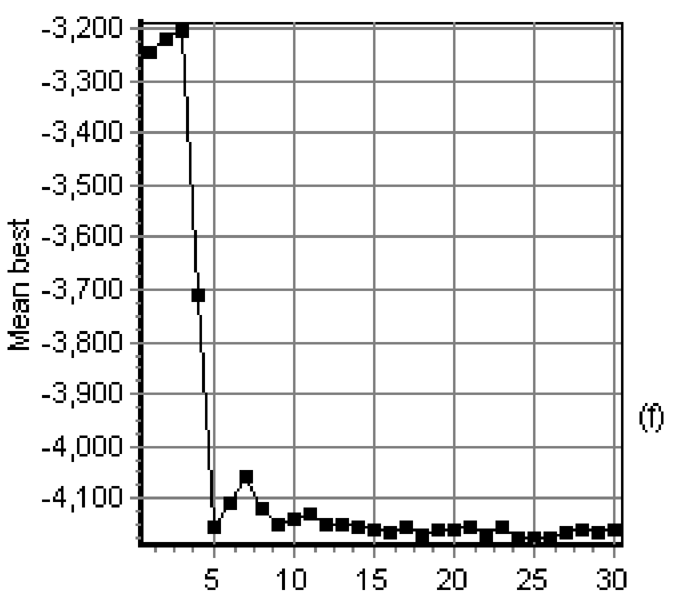

Mean best
(f)

Table 6 Parameters used by AREA in Experiment 3.

Parameter	Value
Number of mutations / chromosome	2
Number of generations	5000
MAX_HARMFUL_MUTATION	3

Having in view Figure 4 we can see the supremacy of the multi-alphabet system over the single - alphabet system. Using multiple alphabets for solution encoding significantly improves the search quality. Using more than 5 alphabets seems to be enough for most of the considered test problems. Using more alphabets (more than 20) does not significantly improve the solution.

4.5 Experiment 4

The relationship between the MAX_HARMFUL_MUTATIONS parameter and the average of the best individual values in the last population is analyzed in this experiment. The parameters used by AREA are given in Table 7.

Table 7 Parameters used by AREA for Experiment 4.

Parameter	Value
Number of alphabets	31
Number of mutations/chromosome	2
Number of generations	5000

The results of this experiment are depicted in Figure 4.

Based on this picture we can see that it is difficult to answer to the question "*What is the optimal value for the MAX_HARMFUL_MUTATIONS parameter*?". It seems that none of the considered values for this parameter is the best for all the test functions. But if the value of this parameter is high the number of alphabets changing is small. In these cases the process behaves like an standard (1 + 1) ES – that does not change the individual representation. If the value for MAX_HARMFUL_MUTATIONS is equal to or greater than the number of generations the initial alphabet will never be changed and the search will be as for the well-known (1+1) ES process.

5 Discussions

Several important issues regarding the AREA representation are discussed in this section.

The unanimously accepted way of applying the mutation operator to chromosomes represented as string of genes is by traversing the chromosome gene by gene and mutating each of these with a mutation probability p_m.

The AREA chromosome is shorter when higher alphabets are used. For instance, a 30 bits chromosome has only 6 digits when the alphabet 32 is used. Thus, when performing mutation by traversing the chromosome, the time AREA takes, using the alphabet 32, is one fifth of the time required for the mutation of a bit string chromosome. That is a rather important way of speeding-up.

Binary bits are grouped in undivided sequences by using higher alphabets. For instance when the alphabet 32 is used, bit sequences of length 5 are represented as a single digit (between 0 and 31). Mutating a digit of such an AREA chromosome actually means mutating the corresponding 5 bits sequence. In that case the mutations, rather numerous (maximum 5 mutations / chromosome as many – if one mutation / chromosome is used when the chromosome is represented over the alphabet 32) are not homogenously distributed over the entire chromosome. They would be so if the mutation operator were applied over the bit string. Thus the good results obtained by using AREA may be connected to this way of applying the mutation operator.

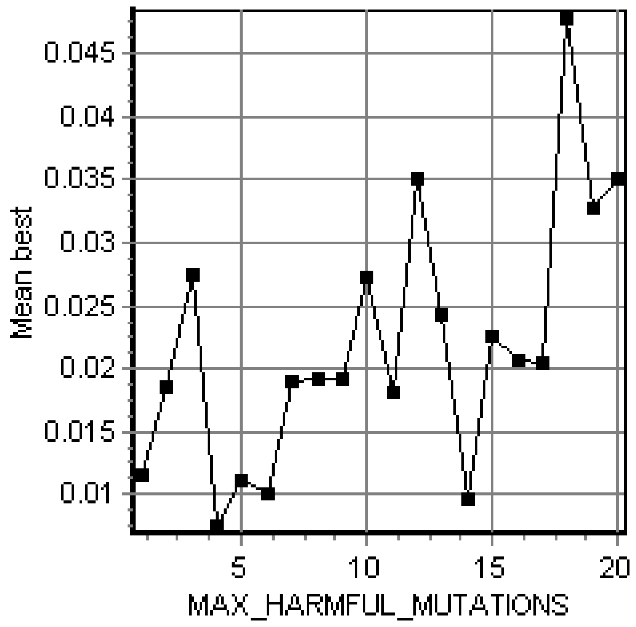
This kind of mutation is in full agreement to process from nature (if a DNA nucleotide is affected by mutation (by radiation, for instance) the neighboring nucleotides have an increased probability of being mutated [9].

At first sight, AREA seems to be a special case of dynamic changing of the mutation probability parameter (but AREA is more than this). The AREA mutation probability is fixed (i.e. one mutation / chromosome) but changing the representation to a higher alphabet generates greater changes as if the mutation probability were changed. In these conditions it is interesting to compare AREA with others techniques that change / adapt the mutation probability during the search process.

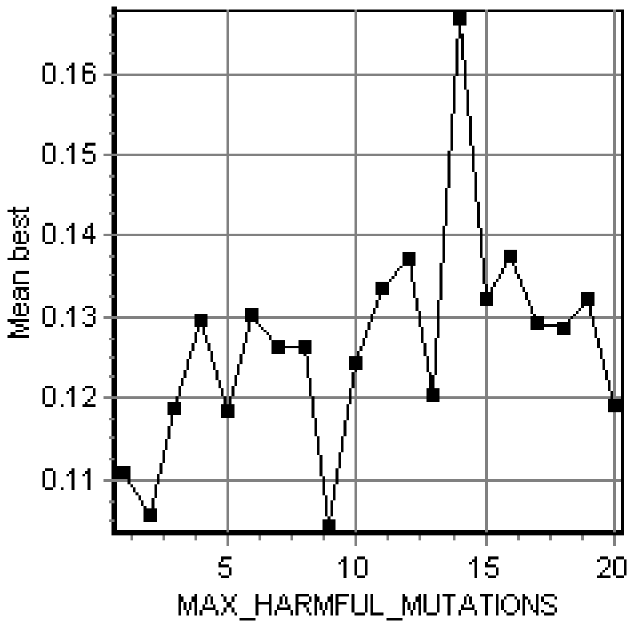
6 Conclusions and Further Work

Taking into account the No Free Lunch Theorems (NFL) (see [16]), we cannot say that AREA is better than other evolutionary algorithms for all of the test problems. Indeed, several cases where other evolutionary algorithms used for comparison are better than AREA have been successfully identified. However, AREA significantly outperforms the standard evolutionary algorithms on the well-known difficult (multimodal) test functions. This advantage of AREA makes it very suitable for real-world applications where we have to deal with highly multimodal functions.

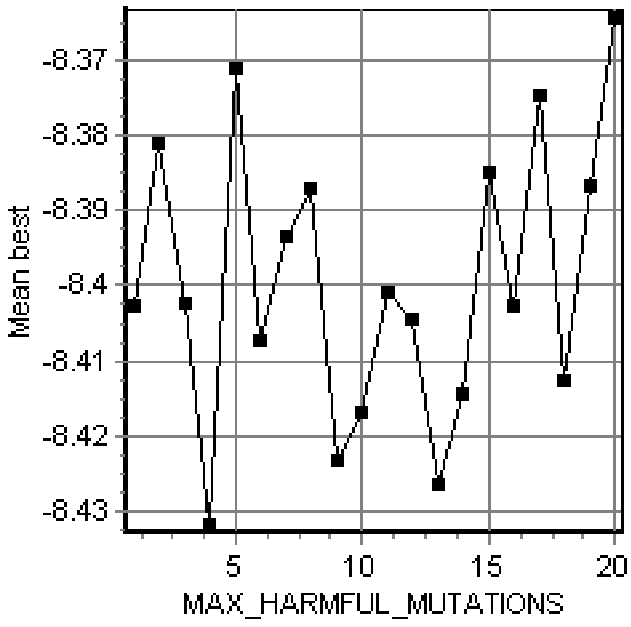
Had only one base been used for solution encoding the gain of AREA over standard ES would have been minimal. Thus, the AREA individuals use a dynamic system of alphabets that may be changed during (and without halting) the search process. If an individual gets stuck in a local optimum - from where it is not able to "jump"-, the individual representation is changed, hoping that this new representation will help the individual


Mean best
MAX_HARMFUL_MUTATIONS

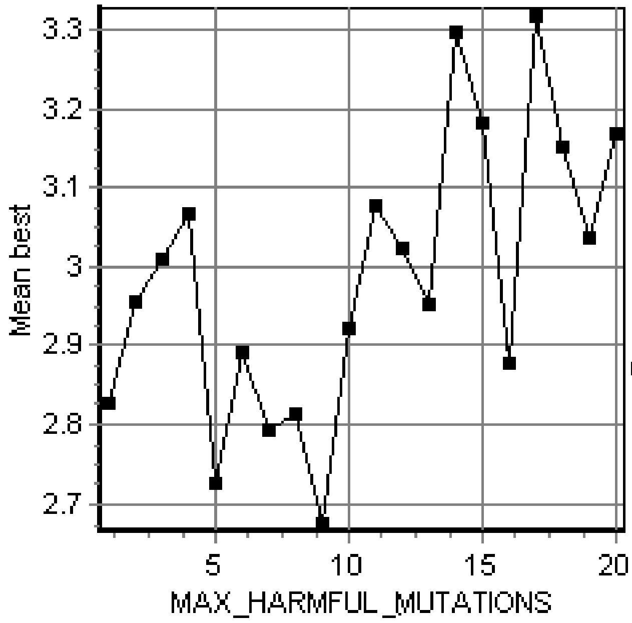
(a)


Mean best
MAX_HARMFUL_MUTATIONS

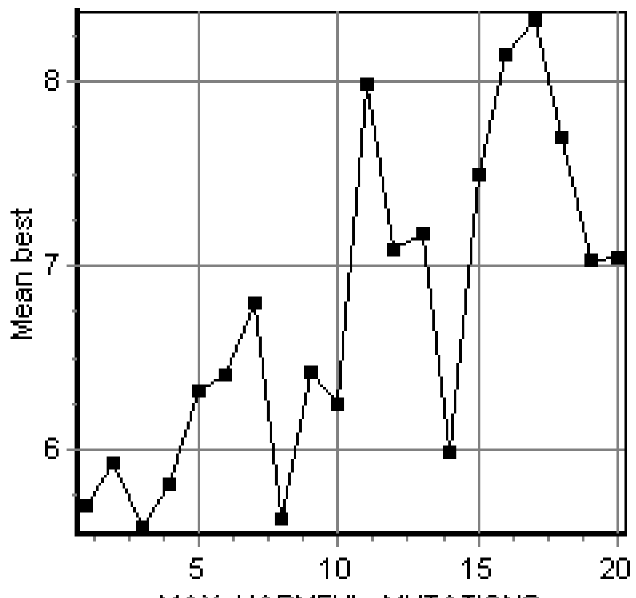
(b)


Mean best
MAX_HARMFUL_MUTATIONS

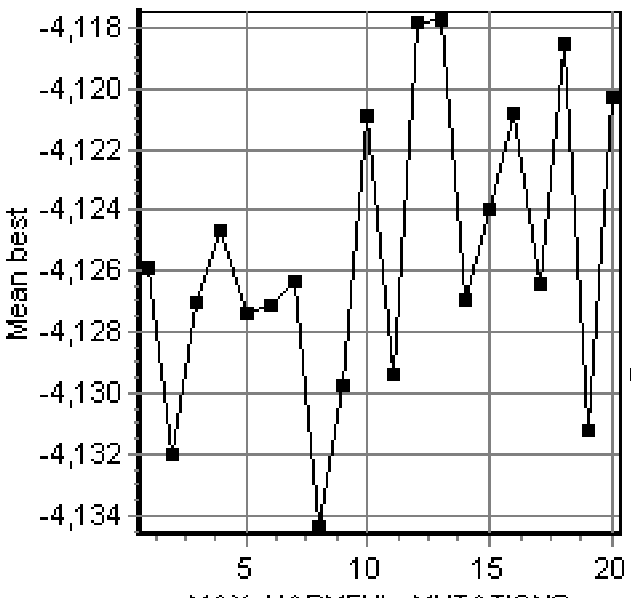
(c)


Mean best
MAX_HARMFUL_MUTATIONS

(d)


Mean best

(e)


Mean best

(f)

to escape from the current position and to explore farther and more efficiently the search space.

The numerical experiments proved that the ability of changing the alphabets when needed is essential. The blind alphabets changes employed by SMES and DRES have a considerable lower ability of converging towards the optimal solution when compared to AREA. Therefore the proposed encoding ensures an efficient exploration of the search space.

The AREA technique could be adopted for other evolutionary techniques such as GP [1] and GA [7]. Further efforts will be dedicated to the embedding of the AREA representation into others standard evolutionary algorithms.

References

1. Angeline, P.: Two self-adaptive crossover operators for genetic programming. In: Angeline, P.J., Kinnear, K.E.,jr., (Eds.): Advances in Genetic Programming, 2. MIT Press, Cambridge, MA (1995) 89-109.
2. Bäck, T.: Optimal mutation rate in genetic search. In Forest, S. (Ed.): Proc. 5^{th} International conference in Genetic Algorithms. Morgan Kaufmann, San Mateo, CA (1993) 2-8.
3. Bäck, T., Schütz, M.: Intelligent mutation rate control in canonical genetic algorithms. Ras, Z.W., Michalewicz, M. (Eds.): Foundations on Intelligent Systems. Lectures Notes in Artificial Intelligence, Vol. 1079. Springer, Berlin (1996) 158-167.
4. Booker, L.B.: Improving search in Genetic Algorithms. In Davis, L. (Ed.). Genetic Algorithms and Simulated Annealing. Morgan Kaufmann, San Mateo, CA (1987) 61-73.
5. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transaction on Evolutionary Computation, Vol. (1999) 124-133.
6. Elseth G. D., Baumgardner K. D., Principles of Modern Genetics, West Publishing Company, (1995).
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, New York (1989).
8. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975).
9. Kimura, M., The Neutral Theory of Molecular Evolution. Cambridge University Press (1983).
10. Kingdon, J., Dekker, L.: The shape of space. Technical Report RN/95/23, Intelligent System Lab, Departament of Computer Science, University College London, London (1995).
11. Rechenberg, I.: Evolutions strategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. FrommannHolzboog Verlag, Stuttgart (1973)
12. Schwefel, H.P.: Numerical Optimization of Computer Models. John Wiley, Chichester (1981).
13. Schaffer, J.D., Morishima, A. An adaptive crossover distribution mechanism for genetic algorithms. Grefenstette, J.J. (Ed.): Proc. 2^{nd} International Conference on Genetic Algorithms. Lawrence Erlbaum Associates, Hillsdale, NJ (1987) 3640.
14. Shaefer, C.G.: The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique, in Laurence, Erlbaum (Eds.), ICGA II. Hillsdale, NJ (1987).
15. Spears, W.M.: Adapting crossover in a genetic algorithm. Report AIC92025, Navy Center for Applied Research in Artificial Intelligence, USA (1992)
16. D.H. Wolpert, W. G. Macready, *No free lunch theorems for optimization*, IEEE Transaction on Evolutionary Computation, (1997) 1:67-82.
17. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Transaction on Evolutionary Computation, Vol. 3(2) (1999) 82-102.