Adaptive Representation Evolutionary Algorithm - a New Technique for Single Objective Optimization

Crina Groşan and Mihai Oltean

Department of Computer Science
Faculty of Mathematics and Computer Science
Babeş-Bolyai University, 3400 Cluj-Napoca, Romania
Email: {cgrosan, moltean}@nessie.cs.ubbcluj.ro

Abstract. In this paper, a new technique called Adaptive Representation Evolutionary Algorithm (AREA) is proposed. 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 optimization problems in this paper. An algorithm for single objective optimization by using the AREA technique is presented. Despite its simplicity the AREA algorithm is able to generate a population converging towards optimal solutions. Numerical experiments indicate that the AREA algorithm performs better than other single objective evolutionary algorithms on the considered test functions.

1 Introduction

Adaptive Representation Evolutionary Algorithm (AREA) is similar to Evolution Strategy (ES) technique [12, 13] 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 a dynamic 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 to sustain AREA. The first example resides in human DNA which is, roughly speaking, a string of nucleotides over the alphabet {T (thymine), C (cytosine), G (guanine), A (adenine)}. By contrast, standard evolutionary algorithms use strings over the alphabet {0, 1} which consists of only two values. The four-nucleotides system has been developed under the specific conditions of

the Earth environment. Had different conditions been on Earth, maybe a ten-nucleotide 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 two-nucleotide systems would have 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 ten-nucleotides 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.

Had only one alphabet 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 (NFL) [17] we cannot say that AREA is better than other evolutionary algorithms for all 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 Genetic Algorithms, 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 [11]. 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, 2, 3, 4, 6, 9, 14, 15, 16].

Several numerical experiments with AREA are performed. The test functions are well-known benchmarking problems used to asses the performances of evolutionary algorithms. Most of these functions are highly multimodal employing different difficulties of the search space.

2 AREA Technique

The main idea of this technique is to allow each solution be encoded over a different alphabet [5, 11]. 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 $\{0, 1, ..., B1\}$. 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 $B = \{0, 1,...B1\}$.

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 $B > 10$.

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 ($B = 8$, $B_2 = 10$).

Remark: A mutation generating an offspring worse than its parent is called a *harmful mutation*.

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.

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 necessary. 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.

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 $p_m = 1 / 6 = 0.66$.

2.4 The AREA Algorithm

The AREA algorithm [5] 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} chromosome from P (t). An offspring is obtained.
Set to zero the number of harmful mutations for offspring;
if the offspring is better than the parent
then the offspring is added to $P(t+1)$;
else begin
Increase the number of harmful mutations for current individual;
if the number of harmful mutations for the current individual = MAX_HARMFUL_MUTATIONS

then begin
Change the individual representation;
Set to zero the number of harmful mutations for the current individual;
Add 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 Numerical Experiments

In this section several experiments with AREA technique are performed. In the first experiment the relationship between the number of iterations and the average of the best individual value in the last generation is analyzed. The result obtained by AREA is compared to the results obtained by two others algorithms. Both of them were introduced in [11]. One of them is called Seasonal Model Evolution Strategy (SMES). In this algorithm the alphabet representation of the solution is changed after a fixed number of generations. The other algorithm is called Dynamic Representation Evolution Strategy (DRES). In that algorithm the alphabet of encoding solution is changed at the end of each generation with a fixed probability.

Test functions used in these experiments are well known benchmarking problems used for assessing and comparing the performances of the search algorithms. Due to the space limitations only the results for two test functions taken from [18] 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 and f_2 are described in what follow. We denote by n the number of space variable, 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 [18] and is a widely used multimodal test function. The global minimum of this function is $x^0 = (0,0,\ldots,0)$ and the function's value in this point in $f(x^0) = 0$.

Test function f_2 is the following:

$$f_2(x) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)).$$

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

Test function f_2 is also known as Rastrigin's function and is based on unimodal function 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 $x^0 = (0,0,\ldots,0)$ and the value of the function in this point is $f(x^0) = 0$.

The number of space dimension was set to 30 for each test function. Each algorithm is run 100 times for each test function in each experiment and with any considered parameters. Because individuals do not interact one with each other, populations with a single individual are used in all the experiments. Using larger populations should bring an increase of the performances. The representation precision was chosen in such way 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 AREA individuals are encoded over the alphabet $\{0, 1\}$ (binary strings). One may initializes all the AREA individuals over a randomly chosen alphabet.

Representation precision for first test function is considered 0.0000001 and for the second function is considered 0.000000007.

3.1 Experiment 1

In this experiment the relationship between the number of iterations and the average of the best individual value in the last generation is analyzed. The results are averaged over 100 runs. The parameters used for AREA are: *Number of alphabets* = 31; MAX_HARMFUL_MUTATION = 50 and *Number of mutation / chromosome* = 2. DRES and SMES use the same parameters as AREA uses. 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.

In Fig. 1 the results of this experiment are depicted.

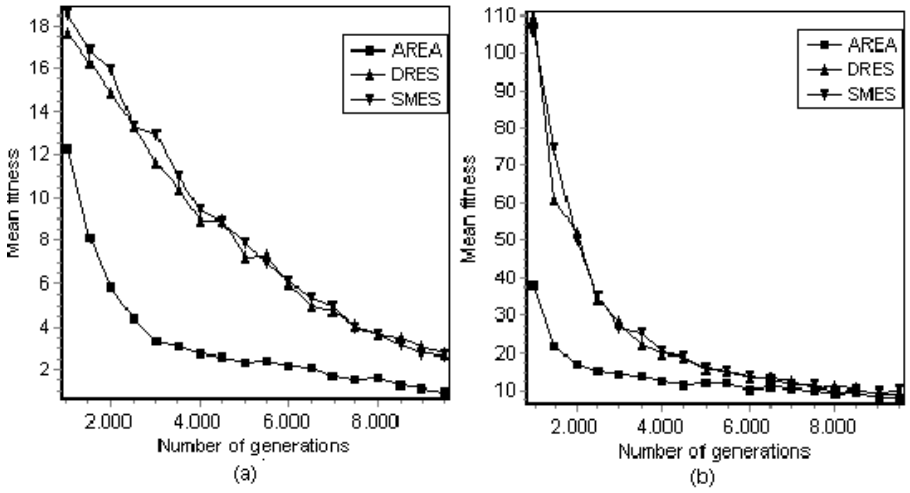

Fig. 1. The relationship between the number of iterations and the average of the best individual value in the last generation over 100 runs. The number of generation varies between 1000 and 10.000. The picture (a) corresponds to the test function f_1 and the picture (b) corresponds to the test function f_2..

AREA significantly outperforms DRES and SMES algorithms. AREA has the best speed of convergence. From this picture we can see the supremacy of Adaptive Representation system over the Dynamic Representation systems.

3.2 Experiment 2

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 results are averaged over 100 runs. Parameters used for AREA are: *Number of mutations / chromosome* = 2; *Number of generations* = 5000 and MAX_HARMFUL_MUTATION = 3. For speed purposes, the test functions are analyzed for 10 dimensions.

In Fig. 2 the results of this experiment are depicted.

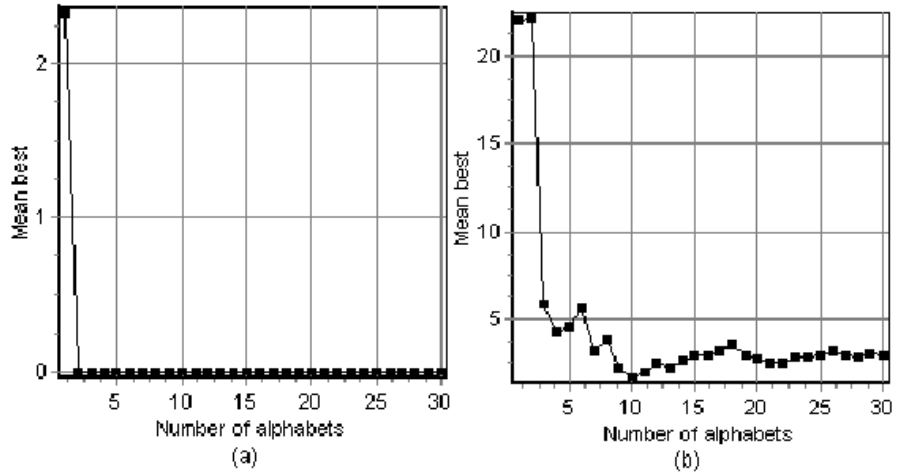

Fig. 2. The relationship between the number of alphabets used to encode chromosomes encoding and the average of the best individual values in the last population over 100 runs. The number of alphabets is varied between 2 and 32. The used alphabets are {0, 1}, {0, 1, 2}, ..., {0, 1, 2,..., 31}. The picture (a) corresponds to the test function f_1 and the picture (b) corresponds to the test function f_2.

From this picture we can see the supremacy of the multi-alphabet system over the single – alphabet system. Using multiple alphabets for solutions encoding significantly improves the search quality. For most of the considered test problems using more than 5 alphabets seems to be enough. Using more alphabets (more than 20, let's say) does not significantly improve the solution.

3.3 Experiment 3

In this experiment the relationship between the MAX_HARMFUL_MUTATIONS parameter and the average of the best individual values in the last population is analyzed. The results are averaged over 100 runs. Parameters used for AREA are: *Number of alphabets* = 31; *Number of mutations / chromosome* = 2 and *Number of generations* = 5000.

In Fig. 3 the results of this experiment are depicted.

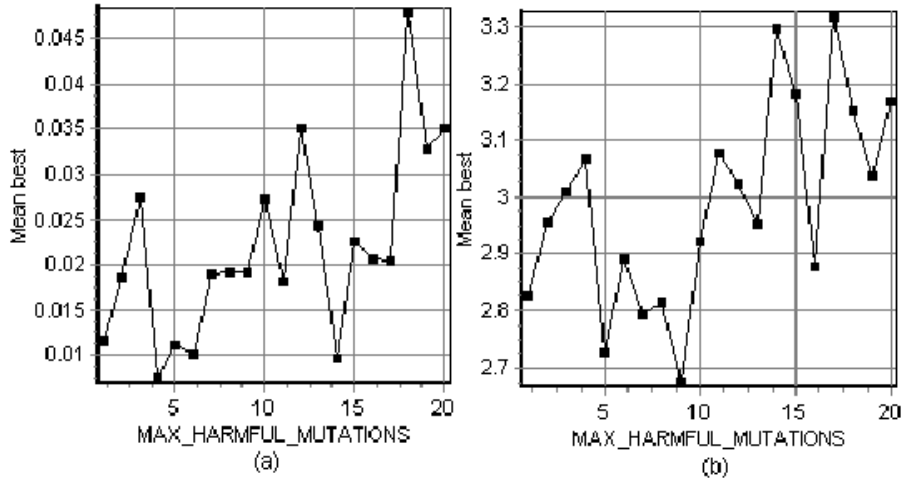

Fig. 3. The relationship between the MAX_HARMFUL_MUTATIONS parameter and the average of the best individual values in the last population over 100 runs. The MAX_HARMFUL_MUTATIONS parameter is varied between 1 and 20. The picture (a) corresponds to the test function f_1 and the picture (b) corresponds to the test function f_2.

From this picture we can see that it is difficult to answer to the question "*Which 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 big the number of alphabets changing is small. In these cases the process behaves like an (1 + 1) ES. If the value for MAX_HARMFUL_MUTATIONS is equal or larger than the number of generations the initial alphabet will never be changed and the search will be as previously described (1+1) ES process.

4 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 neighbouring nucleotides have an increased probability of being mutated.

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.

5 Conclusion and Further Work

A new and efficient evolutionary technique has been proposed in this paper. The AREA individuals are string of genes represented over an alphabet that may be changed during the search process.

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 EP [18], GP [10] and GA [7, 8]. 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. *Proceedings 5th 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. I: Ras Z.W., Michalewicz M. (eds.): *Foundations on Intelligent Systems*. Springer LNCS Vol.1079. Springer, (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. Dumitrescu D., Groşan C., Oltean M.: A New Evolutionary Adaptive Representation Paradigm, Studia Universitas Babeş Bolyai, Cluj-Napoca, Vol.2, (2001) 19-29
6. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter Control in Evolutionary Algorithms. *IEEE Transaction on Evolutionary Computation*, 3:124-139 (1999)
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. Univ. of Michigan Press, Ann Arbor (1975)
9. Julstrom B.A.: What Have you Done for me Lately? Adapting Operator Probabilities in a Steady-State Genetic Algorithm. *Proceeding 6th International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA (1995) 81-87
10. Koza J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, (1992)
11. Kingdon J., Dekker L.: The Shape of Space. Tech Report RN/95/23, Intelligent System Lab, Department of Computer Science, Univ. College London, London (1995)
12. Rechenberg I.: *Evolutions Strategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution.* FrommannHolzboog Verlag, Stuttgart (1973)
13. Schwefel H.P.: *Numerical Optimization of Computer Models.* John Wiley, Chichester (1981)
14. Schaffer J.D., Morishima A. An Adaptive Crossover Distribution Mechanism for Genetic Algorithms. *Proceedings 2nd International Conference on Genetic Algorithms.* Lawrence Erlbaum Associates, Hillsdale, NJ (1987) 36-40
15. Shaefer C.G.: The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique. *Proceedings 2nd International Conference on Genetic Algorithms.* Lawrence Erlbaum Associates, Hillsdale, NJ (1987)
16. Spears W.M.: Adapting Crossover in a Genetic Algorithm. Report AIC92025, Navy Center for Applied Research in Artificial Intelligence, USA (1992)
17. Wolpert D.H., Macready W.G.: No Free Lunch Theorems for Optimization, *IEEE Transaction on Evolutionary Computation*, 1:67-82, (1997)
18. Yao X., Liu Y., Lin G.: Evolutionary Programming made faster. *IEEE Transaction on Evolutionary Computation*, 3(2):82-102 (1999)