In A. Abraham, J. Ruiz-del-Solar, and M. Köppen (eds), *Soft Computing Systems: Design, Management and Applications*, pp. 153-162, IOS Press, Netherlands, 2002.

ANALYZING THE FOUNDER EFFECT IN SIMULATED EVOLUTIONARY PROCESSES USING GENE EXPRESSION PROGRAMMING

CÂNDIDA FERREIRA

Gepsoft, 37 The Ridings, Bristol BS13 8NU, UK
www.gene-expression-programming.com/author.asp
candidaf@gepsoft.com

Gene expression programming is a genotype/phenotype system that evolves computer programs encoded in linear chromosomes of fixed length. The interplay between genotype (chromosomes) and phenotype (expression trees) is made possible by the structural and functional organization of the linear chromosomes. This organization allows the unconstrained operation of important genetic operators such as mutation, transposition, and recombination. Although simple, the genotype/phenotype system of gene expression programming can provide some insights into natural evolutionary processes. In this work the question of the initial diversity in evolving populations of computer programs is addressed by analyzing populations undergoing either mutation or recombination. The results presented here show that populations undergoing mutation recover practically undisturbed from evolutionary bottlenecks whereas populations undergoing recombination alone depend considerably on the size of the founder population and are unable to evolve efficiently if subjected to really tight bottlenecks.

Introduction

Everybody agrees that, by and large, evolution relies on genetic variation coupled with some kind of selection and, in fact, all evolutionary algorithms explore these fundamental processes. However, there is no agreement concerning the best way to create genetic variation, with researchers divided between mutation and recombination [1, 4, 6, 7, 8, 9, 12]. This fact *per se* is extremely revealing, suggesting that existing artificial systems are fundamentally different from one another. Particularly interesting is that the evolvability of the system will depend heavily on the kind of genetic operator used to create variation. And the size and kind of initial populations is closely related to this question.

In all evolutionary algorithms, an evolutionary epoch or run starts with an initial population. Initial populations, though, are generated in many different ways, and the performance and the costs (in terms of CPU time) of different algorithms depend greatly on the characteristics of initial populations. The simplest and less time consuming population is the totally random initial population. However, few evolutionary algorithms are able to use this kind of initial population due not only to structural constraints but also

to the kind of genetic operators available to create genetic modification. The initial populations of gene expression programming (GEP) are totally random and consist of the linear genomes of the individuals of the population [4].

Gene expression programming is a genotype/phenotype system that evolves computer programs of different sizes and shapes (expression trees) encoded in linear chromosomes of fixed length. The genetic encoding used in GEP allows a totally unconstrained interplay between chromosomes and expression trees. This interplay brought about a tremendous increase in performance allowing, consequently, the undertaking of detailed, much needed analysis of fundamental evolutionary processes.

One such analysis is the importance of the initial diversity in evolution. Ernst Mayr hypothesized that, in nature, small groups of founder individuals can give rise to a new species [10, 11]. This is only possible, however, due to the variety of genetic operators that continually introduce genetic modification in the population. The initial diversity question is extremely important in artificial evolutionary systems where founder events are created each time a run starts. And the efficiency of the system will depend, among other things, on how the system deals with evolutionary bottlenecks. Indeed, the evolutionary strategies followed by different artificial evolutionary systems depend greatly on the nature of their respective initial populations.

In GEP, due to the existence of a truly functional and autonomous genome, the implementation of different genetic operators is extremely simplified and Ferreira [4] introduces seven. Furthermore, due to the high efficiency of the algorithm, the performance and the roles of all these operators can be easily and rigorously analyzed revealing the existence of two fundamental types of evolutionary dynamics: non-homogenizing dynamics found in populations undergoing mutation or other non-conservative operators, and homogenizing dynamics found in populations undergoing recombination alone [5]. Therefore, systems with different evolutionary behaviors can be easily simulated in GEP. In this work, the importance of the initial diversity is analyzed in two different systems. The first evolves under mutation and has a non-homogenizing dynamics characteristic of an efficient adaptation. The second evolves under recombination and has a homogenizing dynamics characteristic of poorly evolving systems.

1. Genetic Algorithms

All genetic algorithms use populations of individuals, select individuals according to fitness, and introduce genetic variation using one or more genetic operators. Structurally, genetic algorithms can be subdivided in three fundamental groups: (1) Genetic algorithms with individuals consisting of linear chromosomes of fixed length devoid of complex expression. In these systems, replicators (chromosomes) survive by virtue of their own properties. The algorithm invented by Holland [8] belongs to this group, and is known as genetic algorithm or GA; (2) Genetic algorithms with individuals consisting of ramified structures of different sizes and shapes and, therefore, capable of as-

suming a richer number of functionalities. In these systems, replicators (ramified structures) also survive by virtue of their own properties. The algorithm invented by Cramer [2] and later developed by Koza [9] belongs to this group and is known as genetic programming or GP; and (3) Genetic algorithms with individuals encoded as linear chromosomes of fixed length which are afterwards expressed as ramified structures of different sizes and shapes. In these systems, replicators (chromosomes) survive by virtue of causal effects on the phenotype (ramified structures). The algorithm invented by myself [4] belongs to this group and is known as gene expression programming or GEP.

It is worth emphasizing that GEP shares with GP the same kind of ramified structure, meaning that both systems can be used in the same problem domains. However, due to the crossing of the phenotype threshold [3], gene expression programming is bound to be much more successful, allowing the exploration of new frontiers in evolutionary computation. Below are briefly highlighted some of the differences between GP and GEP.

1.1. Genetic Programming

As simple replicators, the ramified structures of GP are tied up in their own complexity: on the one hand, bigger, more complex structures are more difficult to handle during reproduction and, on the other, the introduction of genetic variation can only be done at the tree level and, therefore, must be done carefully so that valid structures are created. For instance, the tree-specific recombination is practically the only source of genetic variation used in GP for it allows the exchanging of sub-trees and, therefore, always produces valid structures. But the implementation of other operators, like the equivalent of the natural high-performing point mutation, is unproductive as most mutations would have resulted in syntactically incorrect structures.

Obviously, the implementation of other operators such as transposition or inversion raises similar difficulties. In fact, Koza [9] describes two other tree-specific operators, permutation and mutation, but they are seldom used in GP.

1.2. Gene Expression Programming

The phenotype of GEP individuals consists of the same kind of ramified structures used in genetic programming. However, these complex entities are encoded in simpler, linear structures of fixed length – the chromosomes. Thus, there are two main players in GEP: the chromosomes and the ramified structures or expression trees (ETs), the latter being the expression of the genetic information encoded in the former. As in nature, the process of information decoding is called translation. And this translation implies obviously a kind of code and a set of rules. The genetic code is very simple: a one-to-one relationship between the symbols of the chromosome and the functions or terminals they represent. The rules are also very simple: they determine the spatial organization of the functions and terminals in the ETs and the type of interaction between sub-ETs in multigenic systems.

In GEP there are therefore two languages: the language of the genes and the language of ETs. However, thanks to the simple rules that determine the structure of ETs and their interactions, it is possible to infer immediately the phenotype given the sequence of a gene, and *vice versa*. This bilingual and unequivocal system is called *Karva* language. The details of this new language are given in [4].

2. Artificial Evolutionary Systems and the Founder Effect

The question of the initial diversity is pertinent in artificial evolutionary systems for two main reasons. First, the random generation of viable individuals in some complex problems can be a rare event and, in those cases, it would be advantageous if the evolutionary process could get started from one or a few founder individuals; whether this is possible or not, will depend on the modification mechanisms available to the system. And, second, because of this, the kind of mechanism used to create genetic variation becomes of paramount importance. If genetic variation is created by non-homogenizing operators such as point mutation, then populations will be able to adapt and evolve. However, if genetic variation is created by homogenizing operators (recombination), then evolution is either altogether halted when only one founder individual is available or seriously compromised when the number of founder individuals is excessively small.

The importance of the initial diversity in evolution was stressed by E. Mayr in what he called founder effect speciation [10, 11]. This process may be thought of as the establishment of a new population due to a founder event initiated by genetic drift and followed by natural selection. An extreme case of a founder event is the colonization of a previously uninhabited area by a single pregnant female. In nature, besides recombination, other genetic operators are used to create modification and populations that pass through a bottleneck are capable of adaptation, sometimes even originating new species.

Similarly, in artificial evolutionary systems, the capability of founder populations to evolve depends greatly on the kind of mechanism used to create genetic modification. Indeed, if homogenizing operators are the only source of genetic modification, populations will either be unable to evolve efficiently or not at all in the extreme case of only one founder individual.

In this work, different populations of computer programs will be used to analyze the founder effect in evolution. One kind of population uses point mutation as the only source of genetic modification and the other uses only recombination.

3. Setting the System

In order to quantify accurately how different populations respond to the number of actual founders in initial populations, a simple, exactly solved problem must be chosen. This problem must allow the comparison of dissimilarly performing genetic operators, such as the high-performing point mutation and the less powerful recombination. In addition, the populations chosen to make the comparisons must follow different evolution-

ary dynamics so that the results discussed here could be useful not only theoretically but also for understanding the evolutionary strategies chosen by different artificial evolutionary systems.

3.1. General Settings

To analyze the founder effect on populations undergoing either mutation or recombination, the following test sequence was chosen:

$$a_n = 4n^4 + 3n^3 + 2n^2 + n$$

where n consists of the nonnegative integers. This sequence was chosen for three reasons. First, it can be exactly solved by the algorithm and therefore provide an accurate measure of performance in terms of success rate. Second, it requires relatively small populations and relatively short evolutionary times, making the task feasible. And third, it provides sufficient resolution to allow the comparison of dissimilarly performing operators such as mutation and recombination.

In all the experiments, the first 10 positive integers n and their corresponding term were used as fitness cases; the fitness function was based on the relative error with a selection range of 20% and maximum precision (0% error), giving maximum fitness $f_{\max} = 200$ [4]; the selection was made by roulette-wheel sampling coupled with simple elitism; population sizes P of 50 individuals and evolutionary times $G = 100$ generations were used; the success rate of each experiment was evaluated over 100 independent runs; F = {+, -, *, /} and the terminal set T consisted only of the independent variable which was represented by a, giving T = {a}; and six-genic chromosomes of length 78 (head length $h = 6$) linked by addition were used.

In gene expression programming mutation is by far the single most important genetic operator and populations undergoing mutation display non-homogenizing dynamics [5], i.e., the best fitness is always considerably above average fitness and average fitness displays a pronounced oscillatory pattern. Furthermore, mutation is the only operator capable of reaching the performance peak and, for each experiment, this peak can be found. Figure 1 shows the performance peak for populations evolving using the parameters given above. In this case, maximum performance is reached around a mutation rate $p_m = 0.05$. Therefore, this value will be used to study the importance of the initial diversity in non-homogenizing populations.

As for recombination, this operator is the less powerful of all GEP operators and populations undergoing recombination alone display homogenizing dynamics [5], i.e., with time, the best fitness becomes equal to average fitness and populations lose all genetic diversity. Furthermore, it has been shown that the three kinds of GEP recombination (two-point, one-point and gene recombination) perform better at maximum rates of 1.0, being two-point recombination the most powerful of the three recombinational operators and gene recombination the less powerful [5]. However, for the particular settings used in this analysis, when used separately, the three kinds of recombination per-

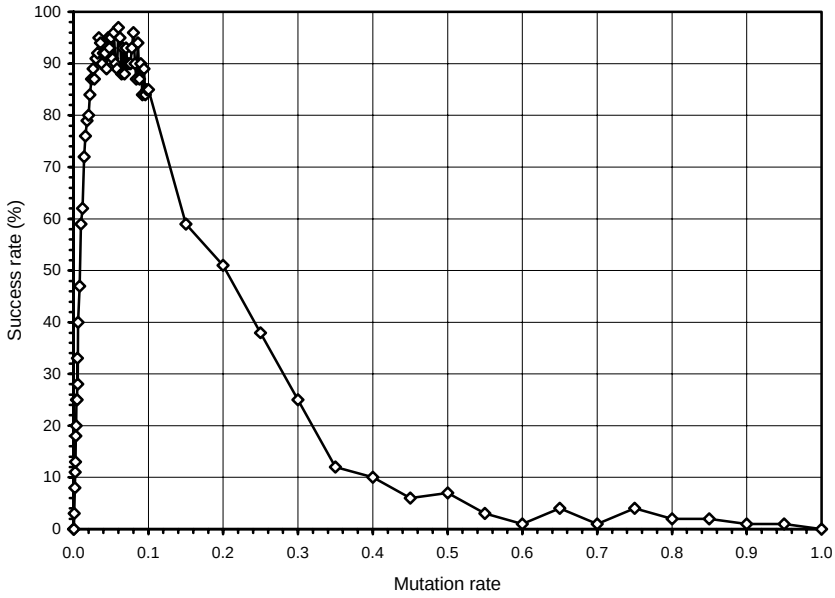

Figure 1. Determining the performance summit using mutation alone.

form so poorly that the three recombinational operators were combined together so that the performance of the algorithm increased a little (Table 1, column 5). As shown in the fifth column of Table 1, in this experiment, the recombination rates of the three recombinational operators are identical and equal to 0.8. Note that the success rate increases slightly comparatively to the individual performances obtained for the recombination operators working separately. So, the dependence of success rate on the number of actual founders in populations undergoing recombination will be analyzed using the general settings shown in the fifth column of Table 1. As will next be shown, the kind of evolutionary dynamics exhibited by these populations is, nonetheless, of the same kind as the homogenizing dynamics characteristic of populations undergoing only one type of recombination at a time.

3.2. Choosing Non-homogenizing and Homogenizing Populations to Study the Founder Effect

In GEP, populations undergoing mutation are characterized by non-homogenizing dynamics where a considerable gap between average and best fitness is maintained throughout the evolutionary history of a population [5]. Furthermore, in this kind of dynamics, the plot for average fitness shows a pronounced oscillatory pattern which reveals the extent of the modifications taking place in the genome of the individuals. Figure 2 shows such a dynamics obtained for a successful run of the experiment summarized in the first column of Table 1.

Table 1. Success rates and parameters for a non-homogenizing system undergoing mutation (Mut) and homogenizing systems undergoing two-point recombination (Rec2P), one-point recombination (Rec1P), gene recombination (RecG), and three different kinds of recombination (RecMix).

	Mut	Rec2P	Rec1P	RecG	RecMix
Number of runs	100	100	100	100	100
Number of generations	100	100	100	100	100
Population size	50	50	50	50	50
Number of fitness cases	10	10	10	10	10
Head length	6	6	6	6	6
Number of genes	6	6	6	6	6
Chromosome length	78	78	78	78	78
Mutation rate	0.05	--	--	--	--
Two-point recombination rate	--	1.0	--	--	0.8
One-point recombination rate	--	--	1.0	--	0.8
Gene recombination rate	--	--	--	1.0	0.8
Selection range	20%	20%	20%	20%	20%
Precision	0%	0%	0%	0%	0%
Success rate	96%	0.04%	0.03%	0.0%	0.13%

On the other hand, populations undergoing recombination alone have homogenizing dynamics [5]. In these populations, the gap between average and best fitness is considerably smaller and, with time, tends to disappear completely. Obviously, when this happens all the individuals in the population have the same genetic makeup and populations become stagnant and incapable of adaptation. Also important is the fact that the plot for average fitness does not show such dramatic oscillations as observed in populations with non-homogenizing dynamics. Note also that, in these systems, populations evolve very inefficiently (see Table 1).

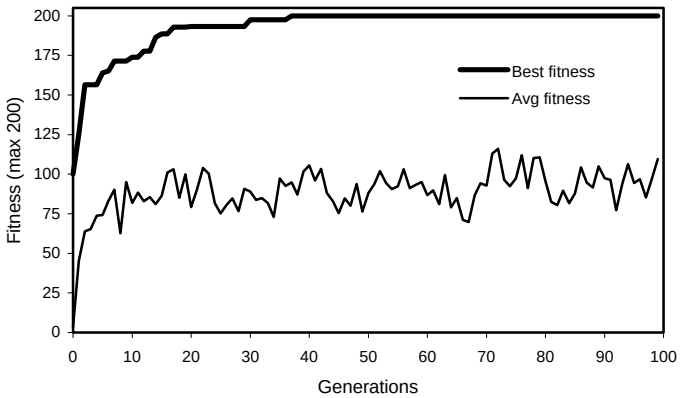

Figure 2. Evolutionary dynamics characteristic of non-homogenizing systems. In this case, the population evolved under a mutation rate of 0.05. Note the oscillatory pattern on average fitness and the wide gap between best and average fitness.

The evolutionary dynamics presented in Figure 3 was obtained for populations subjected to three different kinds of recombination simultaneously (Table 1, column 5). Notwithstanding, these populations exhibit the same homogenizing effect described for populations undergoing only one type of recombination at a time. This further reinforces the hypothesis that recombination is conservative and, therefore, plays a major role at maintaining the status quo [5]. Note that, in this particular case, by generation 54 the

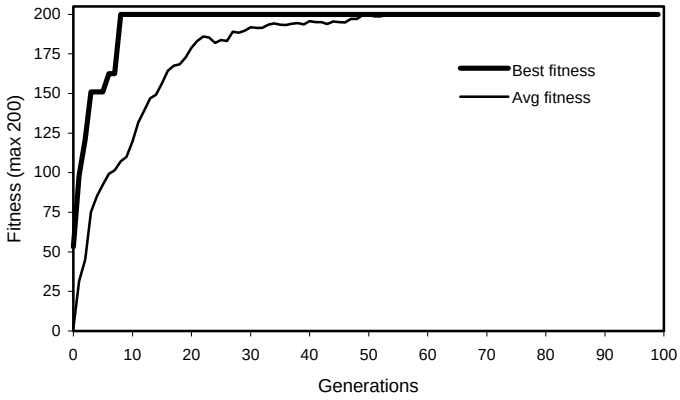

Figure 3. Evolutionary dynamics characteristic of homogenizing populations undergoing recombination. The rates of the three recombination operators used (two-point, one-point and gene recombination) were identical and equal to 0.8. Note the absence of dramatic oscillations on average fitness and that average fitness increases consistently until the complete loss of genetic diversity.

plot for average fitness meets the plot for best fitness and all individuals become genetically identical. This might be seen as a good thing especially if all the individuals would have become equal and perfect. Recall, however, that in complex real-world problems, as in nature, perfection is always a step further ahead. The disadvantages of such an evolutionary strategy, however, become evident when average fitness reaches best fitness before a perfect or good solution is found. Figure 4 shows such a case where the population stabilized on a mediocre solution. In this case, after generation 86 adaptation becomes impossible because all individuals share the same genetic makeup. Indeed, the small success rates typical of populations undergoing recombination alone (see Table 1, for instance) indicate that, most of the times, homogenizing populations converge before finding a good solution because they became irrevocably stuck in some local point, not necessarily optimal.

It is worth noticing that in the experiments summarized in Table 1, totally random initial populations were used and, therefore, the number of viable individuals in those initial populations was not controlled. In the next section it is shown how the number of viable individuals in initial populations can be rigorously controlled in order to analyze the founder effect in artificial evolutionary systems.

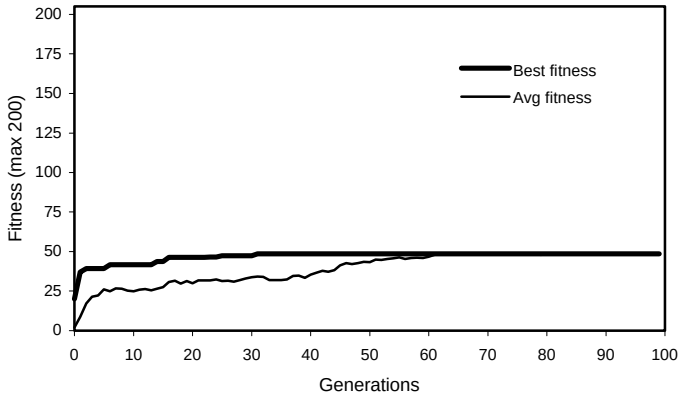

Figure 4. Convergence on a mediocre solution in homogenizing populations undergoing recombination alone. The parameters are exactly the same as in Figure 3.

4. Analyzing the Founder Effect in Simulated Evolutionary Processes

The question of the initial diversity is one of interest in artificial evolutionary systems as a considerable amount of computational resources could go into guaranteeing the adequate initial diversity for populations to evolve efficiently. Here, two different systems will be compared: one that relies on mutation and has a non-homogenizing evolutionary dynamics and another that relies exclusively on crossover and, therefore, has a homogenizing dynamics.

For this analysis, a smaller, totally random "initial" population (founder population) composed of a certain number of viable individuals is created. That is, the run only starts when all the members of the founder population are viable, that is, have positive fitness. These founder individuals are afterwards selected and reproduced, leaving as many descendants as the actual population size *P*.

As shown in Figure 5, for non-homogenizing populations there is no correlation between success rate and the initial diversity. Indeed, due to the constant introduction of genetic modification in the population, in non-homogenizing populations, after a certain time, the founder effect is completely erased and populations evolve, as usual, efficiently.

However, a very different situation happens in populations where crossover is the only source of genetic diversity and the evolutionary dynamics are homogenizing in effect. In these cases, there is a strong correlation between success rate and initial diversity. Note that populations evolve poorly under recombination, being practically incapable of adaptation in the cases where only 2-5 founder individuals are used (obviously, for cases with only one founder, homogenizing populations are altogether incapable of adaptation). It is worth emphasizing that, in these systems, even when the size of the founder population is equal to *P*, the success rate is significantly smaller than in populations undergoing mutation, with only one viable individual in the founder population.

Also worth considering is that the computational resources required to guarantee the

Figure 5. Dependence of success rate on the size of the founder population in non-homogenizing populations undergoing mutation alone (mutation rate equal to 0.05) and homogenizing populations undergoing recombination alone (two-point, one-point and gene recombination rates all equal to 0.8).

creation of large founder populations are very expensive. Thus, systems such as GEP capable of evolving efficiently with minimal initial diversity are most advantageous. Furthermore, for some complex problems like, for instance, the discovery of cellular automata rules for the density-classification task [4], it is very difficult to generate randomly a viable individual, even a mediocre one, to start the run. In those cases, systems like GEP can use this individual as founder and continue from there, whereas systems relying on recombination alone will be stuck for a long time before they gather momentum. In GEP, due to the varied set of genetic operators available such as the high-performing point mutation and transposition, there is no need for large founder populations because as long as one viable individual is randomly generated in the initial population the evolutionary process can get started.

5. Conclusions

The question of the initial diversity in artificial evolutionary systems was addressed using gene expression programming. Due to the varied set of genetic operators and the high efficiency of the algorithm, it was possible to compare dissimilarly performing systems such as systems evolving under mutation alone and systems undergoing only recombination. As most existing artificial evolutionary systems rely either on mutation or recombination, this analysis can help understand the different evolutionary strategies followed by each system.

The results obtained in this work show that, on the one hand, systems using the high-performing mutation operator are not only more efficient but also capable of adaptation under extreme evolutionary bottlenecks. In fact, these systems show no correlation be-

tween the size of the founder population and success rate. Consequently, in the course of a run, this kind of system is never caught in evolutionary cul-de-sacs and, therefore, evolves without end.

On the other hand, systems relying on recombination alone not only perform poorly but also are unable to adapt and evolve when populations pass through a really tight bottleneck. Consequently, these systems not only are useless whenever only one viable individual is available to start an evolutionary epoch but also frequently become irrevocably stuck at evolutionary cul-de-sacs the system itself creates. Because of this, it is mandatory that these systems guarantee a high level of genetic diversity in initial populations for one thing, and for another, the population sizes in these systems must be huge in order to prevent evolutionary cul-de-sacs from happening. Obviously, these systems are highly expensive and impractical.

References

[1] Banzhaf, W., Genotype-Phenotype-Mapping and Neutral Variation – A Case Study in Genetic Programming. In Y. Davidor, H.-P. Schwefel, and R. Männer, eds., *Parallel Problem Solving from Nature III, Lecture Notes in Computer Science*, 866: 322-332, Springer-Verlag, 1994.

[2] Cramer, N. L., A Representation for the Adaptive Generation of Simple Sequential Programs. In J. J. Grefenstette, ed., *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 183-187, Erlbaum, 1985.

[3] Dawkins, R., *River out of Eden*. Weidenfeld and Nicolson, 1995.

[4] Ferreira, C., 2001. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. *Complex Systems*, 13 (2): 87-129.

[5] Ferreira, C., Mutation, Transposition, and Recombination: An Analysis of the Evolutionary Dynamics. In H. J. Caulfield, S.-H. Chen, H.-D. Cheng, R. Duro, V. Honavar, E. E. Kerre, M. Lu, M. G. Romay, T. K. Shih, D. Ventura, P. P. Wang, Y. Yang, eds., *Proceedings of the 6th Joint Conference on Information Sciences, 4th International Workshop on Frontiers in Evolutionary Algorithms*, 614-617, Research Triangle Park, North Carolina, USA, 2002.

[6] Fogel, D. B. and J. W. Atmar, 1990. Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems. *Biological Cybernetics* 63: 111-114.

[7] Fogel, L. J., A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*, New York, Wiley Publishing, 1966.

[8] Holland, J. H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975 (second edition: MIT Press, 1992).

[9] Koza, J. R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA: MIT Press, 1992.

[10] Mayr, E., Change of Genetic Environment and Evolution. In J. Huxley, A. C. Hardy, and E. B. Ford, eds., *Evolution as a Process*, 157-180, Allen and Unwin, London, 1954.

[11] Mayr, E., *Animal Species and Evolution*, Harvard University Press, Cambridge, Massachusetts, 1963.

[12] Rechenberg, I., *Evolutionsstrategie: Optimierung Tecnhischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973.