

# Offspring Selection: A New Self-Adaptive Selection Scheme for Genetic Algorithms

M. Affenzeller, S. Wagner  
Institute for Formal Models and Verification  
Johannes Kepler University, Linz, Austria  
E-mail: {michael, stefan}@heuristicalab.com

## Abstract

In terms of goal orientedness, selection is the driving force of Genetic Algorithms (GAs). In contrast to crossover and mutation, selection is completely generic, i.e. independent of the actually employed problem and its representation. GA-selection is usually implemented as selection for reproduction (parent selection). In this paper we propose a second selection step after reproduction which is also absolutely problem independent. This self-adaptive selection mechanism, which will be referred to as offspring selection, is closely related to the general selection model of population genetics. As the problem- and representation-specific implementation of reproduction in GAs (crossover) is often critical in terms of preservation of essential genetic information, offspring selection has proven to be very suited for improving the global solution quality and robustness concerning parameter settings and operators of GAs in various fields of applications. The experimental part of the paper discusses the potential of the new selection model exemplarily on the basis of standardized real-valued test functions in high dimensions.

## 1 Introduction

By considering crossover as their main reproduction operator, Genetic Algorithms (GAs) take an approach which is fundamentally different to neighborhood-based search techniques. The most essential difference is given by the fact that recombination is able to combine properties of individuals from different regions of the search space. Therefore, provided that the problem representation and the operators are adequate, the advantage of applying GAs to hard optimization problems lies in their ability to search wider regions of the solution space than heuristic methods based upon neighborhood search do. Nevertheless, also GAs are frequently faced with a problem which, at least in its impact, is quite similar to the problem of stagnating in a local optimum. This drawback, called premature convergence in the terminology of GAs, occurs when the population of a GA reaches

such a suboptimal state that the genetic operators can no longer produce offspring which outperform their parents (e.g. [1]).

The main aim of this paper is to find, analyze and improve new generic theoretical concepts for avoiding or at least retarding premature convergence in a generic way:

The basic approaches for retarding premature convergence discussed in GA literature aim to maintain genetic diversity. The most common techniques for this purpose are based upon preselection [2], crowding [3], or fitness-sharing [4]. The main idea of these techniques is to maintain genetic diversity by the preferred replacement of similar individuals [2], [3] or by the fitness-sharing of individuals which are located in densely-populated regions [4]. While methods based upon [3] or [4] require some kind of neighborhood measure depending on the problem representation, [4] is additionally quite restricted to proportional selection. Moreover, these techniques have the common goal to maintain genetic diversity which is very important in natural evolution where a rich gene pool is the guarantor in terms of adaptiveness w.r.t. changing environmental conditions. In case of artificial genetic search as being performed in Genetic Algorithm the optimization goal does not change during the run of a GA and the fixing of alleles of a global optimal solution is desirable in the same manner as the erasement of alleles which are definitely not part of a good solution, i.e. we claim that pure diversity maintenance mechanisms as suggested in [2], [3], or [4] do not support goal-oriented genetic search w.r.t the locating of global optimal solutions.

A very essential question about the general performance of a GA is, whether or not good parents are able to produce children of comparable or even better fitness (the building block hypothesis implicitly relies on this). In natural evolution, this is almost always true. For Genetic Algorithms this property is not so easy to guarantee. The disillusioning fact is that the user has to take care of an appropriate coding in order to make this fundamental property hold.

In order to overcome this strong requirement we have developed an advanced selection mechanism which is based on the idea to consider not only the fitness of the parents, in order to produce a child for the ongoing evolutionary process. Additionally, the fitness value of the evenly produced offspring is compared with the fitness values of its own parents in order to decide whether or not the evenly produced offspring is accepted as a member of the next generation. The offspring is accepted as a candidate for the further evolutionary process if and only if the reproduction operator was able to produce an offspring that could outperform the fitness of its own parents. This strategy guarantees that evolution is presumed mainly with crossover results that were able to mix the properties of their parents in an advantageous way. Via these means we are already in a position to attack one of the reasons for premature convergence, namely the loss of relevant genetic information due to improper crossover operators. Furthermore, this strategy has proven to act as a precise mechanism for self-adaptive selection pressure steering.

## 2 Offspring Selection: A New Model for Self-Adaptive Selection Pressure Steering

In principle, the new selection strategy acts in the following way:

The first selection step chooses the parents for crossover either randomly or in any well-known way of Genetic Algorithms like roulette-wheel, linear-rank, or some kind of tournament selection strategy. After having created a new child from the selected parents by crossover and mutation, we introduce a further selection mechanism that considers the success of the apparently applied reproduction. In order to assure that genetic search proceeds mainly with successful offspring, it has to be attempted that the used crossover and mutation operators are able to create a sufficient number of children that surpass their parents' fitness. Therefore, a new parameter, called success ratio ( $SuccRatio \in [0, 1]$ ), is introduced. The success ratio gives the quotient of the next population members that have to be generated by successful mating in relation to the total population size. We define that a child is successful if its fitness is better than the fitness of its parents, whereby the meaning of 'better' has to be explained in more detail: is a child better than its parents, if it surpasses the fitness of the weaker, the better, or is it in fact some kind of mean value of both?

For this problem, we claim that an offspring only has to surpass the fitness value of the worse parent in order to be considered as 'successful' at the beginning, while as evolution proceeds the child has to be better than a fitness value continuously increasing between the

fitness of the weaker and the better parent. As in the case of Simulated Annealing, this strategy gives a wider search at the beginning, whereas at the end of the search process this operator acts in a more and more directed way. Having filled up the claimed ratio ( $SuccRatio$ ) of the next generation with successful individuals according to the above meaning, the rest of the next generation ( $(1 - SuccRatio) \cdot |POP|$ ) is simply filled up with individuals randomly chosen from the pool of individuals that were also created by crossover but did not reach the success criterion. The actual selection pressure  $ActSelPress$  at the end of a single generation is defined by the quotient of individuals that had to be considered until the success ratio was reached, and the number of individuals in the population in the following way:

$$ActSelPress = \frac{|POP_{i+1}| + |POOL|}{|POP|}$$

If it is more difficult to achieve improvements at a certain stage of genetic search, more selection pressure is needed and this GA variant is able to handle this in a self adaptive way by creating a greater number of offspring as candidates for the next generation. If, on the other hand, evolutionary progress is easier to achieve, this algorithm is able to operate under less selection pressure. The amount of required selection pressure also depends on the actually employed operators: If a certain crossover operator is appropriate for a concrete problem representation, it is easier to generate new offspring which are able to outperform their own parents and less selection pressure is needed. What is really rather remarkable is the fact that also provably worse operators [5] are able to achieve high-quality results as long as at least 'sometimes' advantageous crossover result are produced – the unprofitable crossover results are not considered for the ongoing evolutionary process anyway. Corresponding results for the path representation of the TSP, which show that MPX crossover for example is able to achieve results comparable to ERX crossover or OX crossover (when being equipped with our self-adaptive selection pressure model), are reported in [6].

An upper limit of selection pressure ( $MaxSelPress$ ) defines the maximum number of offspring considered for the next generation (as a multiple of the actual population size) that may be produced in order to fulfill the success ratio. With a sufficiently high setting of ( $MaxSelPress$ ), this new model also functions as a precise 'detector heuristics' for premature convergence:

*If it is no longer possible to find a sufficient number ( $SuccRatio \cdot |POP|$ ) of offspring outperforming their own parents even if ( $MaxSelPress \cdot |POP|$ ) candidates have been generated, premature convergence has occurred.*

### 3 Empirical Analysis on the Basis of Standardized Test Functions

All results of this section are achieved with the HeuristicLab prototyping and development environment (described in ref. 7). The programs are written in the C# programming language using the Microsoft .NET framework 1.1. All values presented in the following tables are the best resp. average values of thirty independent test runs executed for each test case. The average number of evaluated solutions gives a quite objective measure of the computational effort.

In addition to a detailed empirical analysis of standardized TSP benchmarks in higher dimensions (as stated in ref. 6), this paper discusses the performance of the newly introduced theoretical concepts for some commonly used real valued test functions in different dimensions. The main purpose of this additional analysis is to underpin the generality of the newly introduced concepts and methods. As no problem specific knowledge or local search is involved, the identical algorithm can be applied as in the tests for combinatorial optimization problems like the TSP - just the problem representation and the crossover and mutation operators have to be exchanged with standard operators known from GA literature for real valued encoding (e.g. ref. 8).

The test functions that are used in this section have been designed by several authors for analyzing and comparing different optimization algorithms. Most of the test functions have especially been designed to detect weak points of the different methods. In order to demonstrate the capability of offspring selection for achieving high quality solutions for very difficult optimization problems in a general way, we have selected the following test functions that are considered to be more difficult and whose degree of difficulty can be scaled up by increasing the dimension of the search space due to the exponentially increasing number of local minima (except the unimodal Rosenbrock function whose difficulty is given by the extremely flat region around the global optimum).

- **The n-dimensional Rosenbrock function:**

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

for  $-2.048 \leq x(i) \leq 2.048$  with a global minimum of  $f(\vec{x}) = 0$  at  $\vec{x} = (1, 1, 1, \dots, 1)$ .

- **The n-dimensional Rastrigin function:**

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

for  $-5.12 \leq x(i) \leq 5.12$  with a global minimum of  $f(\vec{x}) = 0$  at  $\vec{x} = (0, 0, 0, \dots, 0)$ .

- **The n-dimensional Schwefel function (Sine Root):**

$$f(\vec{x}) = 418.982887272433 \cdot n + \sum_{i=1}^n -x_i \sin(\sqrt{|x_i|})$$

for  $-500 \leq x(i) \leq 500$  with a global minimum of  $f(\vec{x}) = 0$  at  $\vec{x} = (c, c, c, \dots, c)$  with  $c = 420.968746453712$ .

- **The n-dimensional Ackley function:**

$$f(\vec{x}) = 20 + e - 20 \cdot e^{-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}} - e \frac{1}{n} \sum_{i=1}^n \cos(2 \cdot \pi \cdot x_i)$$

for  $-32.768 \leq x(i) \leq 32.768$  with a global minimum of  $f(\vec{x}) = 0$  at  $\vec{x} = (0, 0, 0, \dots, 0)$ .

As all considered test functions have a unique global minimum of 0 we state the results of the following tables as the absolute difference between the achieved result and 0. Many contributions interpret a solution to be globally optimal if it is in a certain range (e.g.  $10^{-5}$ ) around the global optimum. For reasons of objectiveness we state the exact values of the achieved solutions as it frequently happens that it still takes plenty of iterations to improve the solution quality from a value of  $10^{-5}$  to  $10^{-20}$  for example. Consequently, a displayed result of 0.000000000 has to be interpreted to be smaller than the lower limit of the C# double data type.

Tab. 1 shows the parameter settings of the GA with offspring selection for the different benchmark test functions.

**Table 1.** Parameter values used in the test runs of the GA with offspring selection for the different test functions

Parameters for the n-dimensional Rosenbrock function	
Population Size	100
Elitism Rate	1
Mutation Rate	0.05
Selection Operator	Roulette
Crossover Operators	Discrete, Average, Heuristic
Mutation Operator	One-position Uniform Mutation
Success Ratio	0.6
Comparison Factor	1.0 (constantly)
Maximum Selection Pressure	30
Parameters for the n-dimensional Rastrigin, Schwefel (Sine Root), and Ackley functions	
Population Size	500
Elitism Rate	1
Mutation Rate	0.05
Selection Operator	Roulette
Crossover Operators	Discrete, Average, Heuristic
Mutation Operator	One-position Uniform Mutation
Success Ratio	0.9
Comparison Factor	0.9 - 1.0
Maximum Selection Pressure	25

Many new theoretical serial and parallel GA concepts are analyzed considering the highly multimodal Rastrigin, Ackley, Griewangk or Schwefel's Sine Root function (e.g. [9], [10], [11], etc.). However, these papers analyze rather low problem dimensions around  $n = 10$  for the Rosenbrock function and about  $n = 20$  or  $n = 30$  for the other test functions. Under this perspective it is quite remarkable that the GA with offspring selection is

**Table 2.** Experimental results achieved for the benchmark test function in various dimensions

Results for the Rosenbrock function			
Problem Dimension	Best	Average	Evaluated Solutions
n=20	$1.24 \cdot 10^{-25}$	$5.70 \cdot 10^{-25}$	2'926'231
n=100	$1.37 \cdot 10^{-8}$	$2.23 \cdot 10^{-5}$	16'672'523
n=500	$2.31 \cdot 10^{-9}$	$1.02 \cdot 10^{-8}$	181'217'124
n=1000	$4.11 \cdot 10^{-8}$	$2.59 \cdot 10^{-7}$	493'559'814
Results for the Rastrigin function			
Problem Dimension	Best	Average	Evaluated Solutions
n=20	0.000000000	0.000000000	4'791'050
n=100	0.000000000	0.000000000	10'718'323
n=500	0.000000000	0.000000000	52'423'489
n=1000	$4.26 \cdot 10^{-9}$	$1.03 \cdot 10^{-7}$	67'145'078
n=5000	$4.35 \cdot 10^{-5}$	$4.32 \cdot 10^{-3}$	167'019'244
Results for the Schwefel (Sine Root) function			
Problem Dimension	Best	Average	Evaluated Solutions
n=20	$-1.34 \cdot 10^{-12}$	$-2.10 \cdot 10^{-11}$	735'411
n=100	0.000000000	0.000000000	3'321'411
n=500	$7.31 \cdot 10^{-9}$	$1.54 \cdot 10^{-4}$	17'311'622
n=1000	$5.21 \cdot 10^{-5}$	$2.76 \cdot 10^{-3}$	34'703'482
n=5000	$4.32 \cdot 10^{-4}$	$5.88 \cdot 10^{-3}$	146'235'943
Results for the Ackley function			
Problem Dimension	Best	Average	Evaluated Solutions
n=20	$4.51 \cdot 10^{-17}$	$7.83 \cdot 10^{-16}$	1'171'012
n=100	$3.11 \cdot 10^{-15}$	$7.16 \cdot 10^{-14}$	5'278'856
n=500	$2.85 \cdot 10^{-13}$	$8.78 \cdot 10^{-9}$	39'433'869
n=1000	$6.31 \cdot 10^{-8}$	$1.26 \cdot 10^{-7}$	95'327'512
n=5000	$3.21 \cdot 10^{-6}$	$7.65 \cdot 10^{-5}$	164'976'566

able to find the global optimum of all those test functions up to a problem dimension of  $n = 1000$  for the Rosenbrock function, and even up to  $n = 5000$  for Rastrigin, Ackley, Griewangk and Schwefel without any additional algorithmic adaptations, i.e. with absolutely the same algorithm as used for the combinatorial optimization problems like the TSP. Just the operators (crossover and mutation) have been replaced by standard crossover operators for real-valued encoding.

#### 4 Conclusion

In this paper an enhanced GA selection concept has been presented and exemplarily tested on some real-valued test function with a scalable degree of difficulty. The proposed GA-based techniques couple aspects from Evolution Strategies (variable selection pressure) with established selection, crossover and mutation concepts known from GA theory in a general and self-adaptive way. Therefore established crossover and mutation concepts may be used analogously to the corresponding GA and offspring selection can be applied in various fields of GA applications.

The self-adaptive selection pressure steering makes sure that the actually used selection pressure is just high enough to successfully guide genetic search. Given an upper limit of selection pressure to be applied, this model also acts as a precise detector heuristics for premature convergence.

Furthermore, the potential of offspring selection for Genetic Programming (GP) applications is analyzed

within the scope of a current research project [12]. First results for the identification of nonlinear structures in mechatronic systems by means of Genetic Programming indicate that offspring selection operates especially well in combination with genetic programming.

#### 5 References

- [1] Fogel, D.B. (1994) An introduction to simulated evolutionary optimization. IEEE Transactions on Neural Networks 5 (1): 3–14
- [2] Cavicchio, D.J. (1970) Adaptive Search using Simulated Evolution. Unpublished doctoral dissertation, University of Michigan, Ann Arbor
- [3] De Jong, K.A. (1975) An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD. Thesis. University of Michigan
- [4] Goldberg, D. E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley Longman
- [5] Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., Dizdarevic, S. (1999): Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review 13: 129–170
- [6] Affenzeller, M., Wagner S. (2004): SASEGASA: A new generic parallel evolutionary algorithm for achieving highest quality results. Journal of Heuristics - Special Issue on New Advances on Parallel Meta-Heuristics for Complex Problems, vol. 10: 239-263
- [7] Wagner S., Affenzeller M. (2004): HeuristicLab - A generic and extensible optimization environment. Proceedings of ICANNGA 2005
- [8] Dumitrescu, D., Lazzarini B, Jain L.C., Dumitrescu, A. (2000): Evolutionary computation. CRC Press
- [9] Potter, M.A., De Jong K. (1994): A cooperative coevolutionary approach to function optimization. In: Parallel Problem Solving from Nature – PPSN III, pp. 249–257
- [10] Hiroyasu, T., Miki M., Hamasaki M, Tanimura, Y. (2000): A new model of distributed genetic algorithm for cluster systems: Dual individual DGA. High Performance Computing, Lecture Notes in Computer Science 1940, pp.374–383
- [11] Takahashi, O., Kita, H., Kobayashi S. (1999): A distance dependent alternation model on real-coded genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics 24(4): 619–624
- [12] Winkler S., Affenzeller M., Wagner S. (2004): Identifying nonlinear model structures using genetic programming techniques. In: Proceedings of the European Meeting on Cybernetics and Systems Research - EM-CSR 2004, pp. 689–694