Strategy Adaptation by Competing Subpopulations

Dirk Schlierkamp-Voosen * and Heinz Mühlenbein

GMD Schloß Birlinghoven
D-53754 Sankt Augustin, Germany

Abstract. The breeder genetic algorithm BGA depends on a set of control parameters and genetic operators. In this paper it is shown that strategy adaptation by competing subpopulations makes the BGA more robust and more efficient. Each subpopulation uses a different strategy which competes with other subpopulations. Numerical results are presented for a number of test functions.

Keywords: breeder genetic algorithm, strategy adaptation, competition, multiresolution search

1 Introduction

Many evolutionary algorithms depend on a set of control parameters. Often the optimal setting of the parameter depends on the particular application. Moreover the optimal control parameters may be different at the start of the run and at the end where the individuals are very similar to each other.

Basically two approaches have been pursuit to solve the above problem. In the first approach some externally specified schedule is used. The schedule may depend for instance on the time, measured in number of generations. This approach is derived from simulated annealing. The temperature is set at a large initial value, then it is continuously reduced. In genetic algorithms this approach has been tried for changing the mutation rate or the selection [4].

In the second approach the mechanisms of evolution itself are used to adapt the control parameters. The adaptation is not driven by an external schedule but by the internal forces of evolution itself. This approach is successfully used in evolution strategies [3]. In evolution strategies the adaptation of control parameters is done in the same manner as the adaptation of the parameters defining the fitness functions [1].

The crucial question of the second approach concerns the level where the adaptation is done. It may be the level of the individual as it is done in evolution strategies. But also the level of subpopulations or the level of populations

* schlierkamp-voosen@gmd.de

In: *Parallel Problem Solving from Nature (PPSN III)*, pages 199–208, Springer, Jerusalem, October 1994

can be used. The level implicitly defines the time interval when the adaptation takes place. If for instance populations are used for adaptation, then a minimum number of generations is needed for evaluating the populations. In contrast, individuals are evaluated after each generation.

In this paper we present an adaptation based on subpopulations. It simulates subpopulations competing for the same food. The outline of the paper is as follows. In section 2 the most important control parameters of the breeder genetic algorithm BGA are summarized. In section 3 the competition scheme is explained. The performance of the competition is shown in section 4 for unimodal functions. In section 5 the efficiency and the robustness of the adaptation by competition is demonstrated for multimodal functions.

2 The BGA for Continuous Parameter Optimization

Let an unconstrained optimization problem be given on a domain $D \subset \mathbb{R}^n$

$$\min(F(\mathbf{x})) \qquad a_i \leq x_i \leq b_i \quad i = 1, ..., n \ . \tag{1}$$

The breeder genetic algorithm **BGA** was designed to solve the above problem [6]. The BGA depends on some control parameters which we summarize shortly. The selection is done by *truncation selection*, also called mass selection by breeders. The $T\%$ best of the individuals are selected as parents and then mated randomly.

Discrete recombination
Let $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ be the parent strings. Then the offspring $\mathbf{z} = (z_1, \ldots, z_n)$ is computed by

$$z_i = \{x_i\} \ or \ \{y_i\} \tag{2}$$

x_i or y_i are chosen with probability 0.5.

BGA mutation
A variable x_i is selected with probability p_m for mutation. The BGA normally uses $p_m = 1/n$. At least one variable will be mutated. A value out of an interval $[-range_i, range_i]$ is added to the selected variable. $range_i$ defines the *mutation range*. It is normally set to 0.5 times the domain of definition of variable x_i.

Given x_i a new value z_i is computed according to

$$z_i = x_i \pm range_i \cdot \delta \tag{3}$$

The + or − sign is chosen with probability 0.5. δ is computed from a distribution which prefers small values. This is realized as follows

$$\delta = \sum_{i=0}^{k-1} \alpha_i \cdot 2^{-i} \quad \alpha_i \in \{0, 1\}$$

k is called the precision constant. Before starting the mutation we set $\alpha_i = 0$. Then each α_i is flipped to 1 with probability $p_\delta = 1/k$. Only $\alpha_i = 1$ contributes

to the sum. On the average there will be just one α_i with value 1, say α_j. Then δ is given by

$$\delta = 2^{-j}$$

This mutation scheme is discrete. For a number of reasons we now use a continuous mutation scheme where δ is computed as follows

$$\delta = 2^{-k \cdot \alpha} \quad \alpha \in [0, 1]$$

The discussion of this scheme is outside the scope of this paper.

BGA line recombination

The BGA line recombination uses components from both, mutation and recombination. It creates new points in a direction given by the two parent points. The placement of the point is done by the BGA mutation scheme. It works as follows: Let $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ be the parent strings with $\mathbf{x}$ being the one with better fitness. Then the offspring $\mathbf{z} = (z_1, \ldots, z_n)$ is computed by

$$z_i = x_i \pm range_i \cdot \delta \cdot \frac{y_i - x_i}{\|\mathbf{x} - \mathbf{y}\|} \tag{4}$$

The $-$ sign is chosen with probability 0.9. The offspring is placed more often in the descending direction.

The rationale behind the three operators is as follows. The BGA mutation operator is able to generate *any* point in the hypercube with center $\mathbf{x}$ defined by $x_i \pm range_i$. But it tests much more often in the neighborhood of $\mathbf{x}$. In the above standard setting, the mutation operator is able to locate the optimal x_i up to a precision of $range_i \cdot 2^{-(k-1)}$. Discrete recombination is a breadth search. It uses the information contained in the two parent points. The BGA line recombination tries new points in a direction defined by the parent points.

In [6] we have proven that a BGA with popsize $N = 1$ (1 parent, 1 offspring, the better of the two survives) using only mutation has approximate *linear order of convergence* for unimodal functions.

Theorem 1. *Given a point with distance* $range_i \cdot 2^{-(k-1)} \leq r \leq range_i$ *to the optimum, then the expected progress* $E(n, r)$ *of the BGA in* n *dimensions is bounded by*

$$\frac{1}{2kn} \leq \frac{E(n, r)}{r} \leq \frac{1}{kn} \tag{5}$$

This theorem shows the following problem. The progress depends on k. The larger k, the smaller the progress. But in order to locate the optimum with a given precision ϵ, the value $range_i \cdot 2^{-(k-1)}$ has to be less than ϵ. Therefore a large k may be necessary.

We show the dependence of the BGA on the precision constant k in Fig. 1. The task is to minimize the hypersphere of dimension $n = 100$.

$$F_0(x) = \sum_i^n x_i^2$$

Note that the best fitness is displayed on a logarithmic scale. The simulations have been done with three precision values $k = 4, 8$, and 16. For $k = 16$ one observes for quite a time the predicted linear order of convergence. Small values of k have a better progress at the beginning, but they are able to locate the best fitness to a certain precision only.

Next we summarize the mathematical properties of discrete recombination. A detailed investigation can be found in [7]. Discrete recombination has also *linear order of convergence* in number of generations versus fitness until near the equilibrium. Equilibrium is defined as all genotypes of the population being equal. The fitness value achieved at equilibrium depends on the size of the population and the truncation selection threshold.

In Fig. 2 three simulation runs are shown. One clearly observes the linear order of convergence until near the equilibrium. The rate of progress is larger for a small truncation threshold, but the population converges to a higher fitness value.

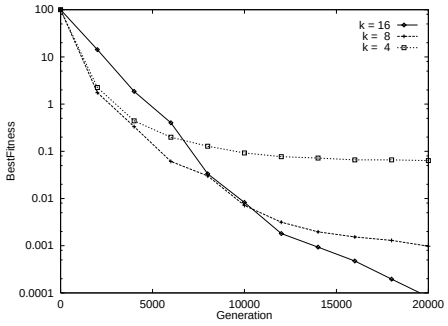

Fig. 1. BGA mutation; hypersphere $n = 100$; precision $k = 16, 8$, and 4.

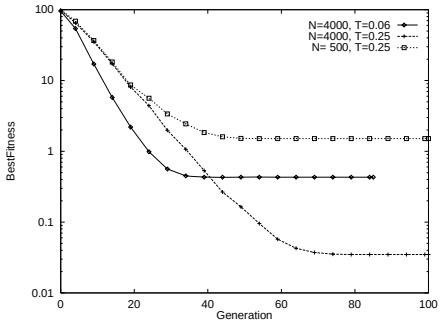

Fig. 2. Discrete recombination, hypersphere $n = 100$. The gradient of the best fitness achieved depends on the truncation threshold T for selection. For $N = 4000$ $T = 0.25$ and $T = 0.06$ was used.

The computational efficiency of mutation and recombination can be compared by changing the abscissa from number of generations to number of function evaluations. It is easily seen that mutation is by far more efficient. This result indicates that recombination is not an efficient search method to determine the minimum of a quadratic function. We will later show that recombination is an important search operator for determining promising search areas of multimodal functions.

The result of the simulations can be summarized as follows: *The efficiency of the BGA mutation operator depends on the precision constant k. The efficiency of discrete recombination depends on the size of the population and the truncation threshold T.*

The question now arises how to combine recombination and mutation in an optimal way and how to automatically control the precision constant k of the

BGA mutation scheme. We will use for the adaptation the concept of competition between subpopulations. This will be described in the next section.

3 Competition between Subpopulations

The adaptation of parameters controlling evolutionary algorithms can be done on different levels, for example the level of the individuals, the level of subpopulations or the level of populations. Bäck et al. [2] have implemented the adaptation of strategy parameters on the level of the individual. The strategy parameters of the best individuals are recombined, giving the new stepsize for the mutation operator in the next generation. Herdy [5] uses competition on the population level. In this case whole populations are evaluated at certain generations. The strategies of the successful populations proliferate, the strategies of populations with bad performance die out. They are replaced by the successful strategies and afterwards modified by genetic operators. There is no exchange of individuals between populations.

Our adaptation lies between these two extreme cases. The competition is done between subpopulations (groups). The total number of all individuals is fixed whereas the size of a single group varies. Our approach simulates the population changes of species which compete for the same food. Well adapted species increase whereas poorly adapted species decrease. This follows Gausse's principle: *'Two species with identical requirements cannot co-exist in a habitat.'* Occasionally individuals with a good fitness migrate to other groups.

The competition requires a *quality criterion* to rate a group, a *gain criterion* to reward or punish the groups, an *evaluation interval*, and a *migration interval*. The evaluation interval gives each strategy the chance to demonstrate its performance in a certain time window. By occasional migration of the best individuals groups which performed badly are given a better chance for the next competition. The sizes of the groups have a lower limit. Therefore no strategy is lost.

The **quality criterion** is based on the fitness of the best individual of the group. To avoid an inefficient oscillation of group sizes we had to extend the quality criterion. For the evaluation information about the last 10 competitions is used. The group with the best individual gets $best\_ind = 1$, for all other groups $best\_ind$ is set to 0.

The following formula describes the quality of group i. $k = 0$ denotes the current competition, $k = 1$ the previous one, etc.

$$quality(i) = \sum_{k=0}^{9} \left(\frac{10 - k}{10} \cdot best\_ind_k(i) \right) \tag{6}$$

The **gain criterion** defines how to modify the population size of each group according to its quality. The size of the group with the best quality increases, all other groups are decreased.

If group i has the best quality, then

$$N_{t+1}^i = N_t^i + \sum_{j=0, j\neq i}^{G-1} \frac{N_t^j}{8} \tag{7}$$

where N_t^i denotes the size of group i and G denotes the number of groups. All other groups are reduced if their size is greater than the minimal size N_{min}.

$$N_{t+1}^j = N_t^j - \frac{N_t^j}{8} \quad j \neq i \tag{8}$$

This gain criterion leads to a fast adaptation of the group sizes. Each group looses the same percentage of individuals.

The **evaluation interval** is normally set to 4, the **migration interval** to 16.

4 Competition for Unimodal Functions

The behavior of the BGA competition scheme can best be explained with the unimodal hypersphere function. In Fig. 3 four groups with different mutation ranges compete. The mutation intervals are defined in Table 1.

Table 1. Range and mutation steps for precision constant $k = 7$ used.

group	range	max. step	min. step
0	5.120000	10.240000	0.1600000
1	0.160000	0.320000	0.0050000
2	0.005000	0.010000	0.0001560
3	0.000156	0.000312	0.0000049

The different mutation ranges define a multiresolution search. Group 0 is doing large mutation steps and group 3 the smallest. The group with the largest range was initialized with 52 individuals, all the other with the minimum popsize of 4. In the rightmost figure one clearly observes the migration interval which was set to $mig = 32$ for reasons of presentation. At these intervals migration takes place. Therefore the best fitness of the groups becomes equal.

In Fig. 4 the distribution of the population sizes of the four groups is shown. First the group with the largest mutation steps dominates, then the group with the second largest mutation steps takes over and so on. The change of the sizes of the groups correspond to the four waves which can be seen in Fig. 3. In Fig. 5 the quality criterion used for the competition is shown.

In Fig. 6 the competition run is compared to a run without competition. The BGA without competition was initialized with a precision constant of $k = 22$. This precision is comparable to the competition run. Until generation 700 both runs are equally effective. Afterwards the competition run is more effective. This behavior is predicted by the theory.

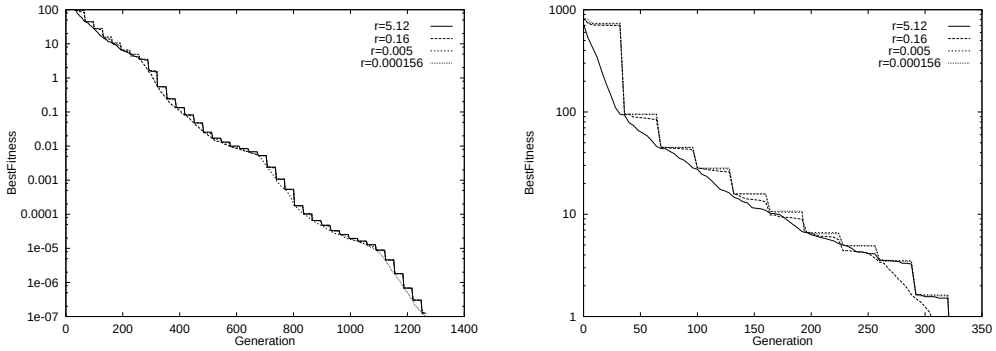

Fig. 3. Competition between 4 groups using different mutation ranges. The precision is $k = 7$.

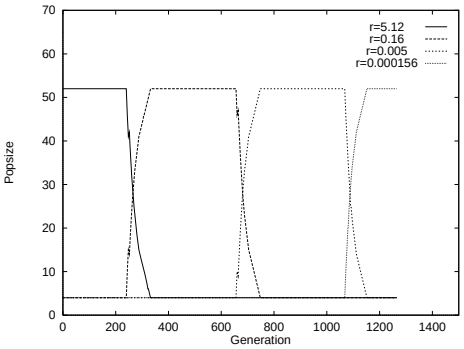

Fig. 4. Distribution of the population sizes; first the group with the largest mutation steps is successful. At the end the group with the smallest mutation steps has taken over.

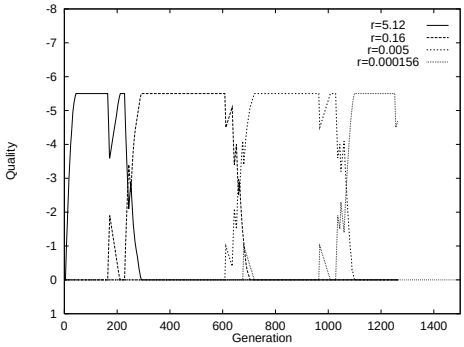

Fig. 5. Quality criterion which affected the variation of the population sizes shown in Fig. 4.

5 Multiresolution Search for Multimodal Functions

The BGA is intended to solve multimodal optimization problems. In [6] we have shown that the standard BGA using mutation and discrete recombination has also linear order of convergence for some of the popular multimodal test functions. The scaling constants are specific to the fitness function and the precision constant k of the BGA. They are less than for unimodal functions. The reason for this behavior can easily be explained. Most of the test functions have a global structure which is similar to the hypersphere. Therefore the multimodality of the function can be considered as noise for the BGA. The multimodality only reduces the probability of creating better offspring.

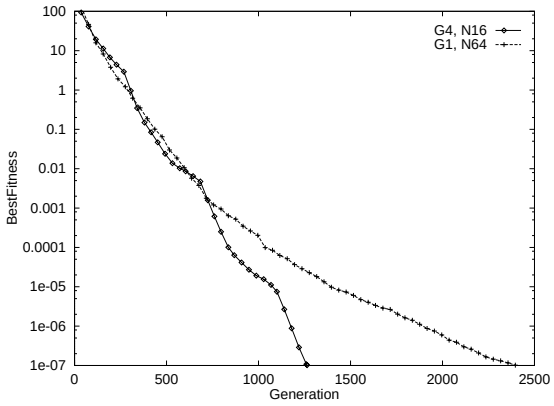

Fig. 6. Competition vs. normal BGA run; For fine tuning competition is more effective. Note the waves of the competition run.

We have pointed out in [6] that the local minima of these test functions are regularly distributed. For these classes of problems discrete recombination is an especially efficient operator. Therefore these test functions are not a challenge for the BGA. For comparisons reason we give some results for the most difficult of these test functions, this is Griewank's function. We will later investigate two functions which we believe are more typical for real life applications.

$$F_8(x) = \sum_{1}^{n} \frac{x_i^2}{4000} - \prod_{1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad -600 \leq x_i \leq 600 \tag{9}$$

In Fig. 7 a competition run and a run without competition is shown for Griewank's function of dimension $n = 100$. Both runs used in addition to mutation discrete recombination. The run without competition is slightly more effective. But the BGA with competition is more robust. The BGA without competition converged in 9 of 10 cases to the second minima.

We will now turn to optimization problems where the local minima are distributed more randomly. In [9] these kind of test functions are also proposed as a common benchmark for global optimization problems. We have used the following test function originally proposed by Rechenberg.

$$F_{12}(x) = \sum_{i=0}^{20} \left((100 - i) \cdot \exp\left(-\sum_{k=1}^{n} \left(\frac{x_k - z_{30 \cdot i + k}}{\sigma} \right)^2 \right) \right) \tag{10}$$

$$\text{where} \quad z_j = (32 \cdot z_{j-1} + 13(i+1)) \bmod 31, \quad z_0 = 1, \quad -100 \leq x_i \leq 100$$

The function consists of 21 exponential mountains, whose positions are randomly distributed. σ is used to vary the shape of the single hills. Due to limitations of space we have to summarize the simulation results. Competition runs

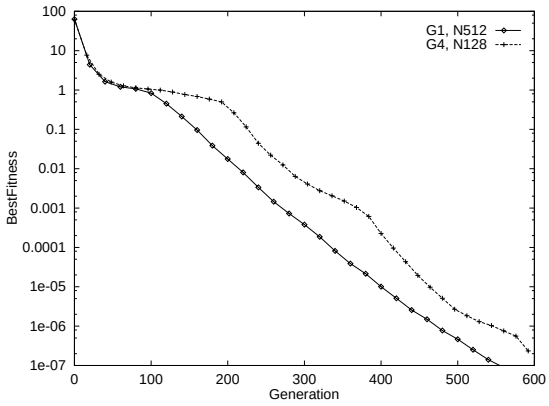

Fig. 7. Griewangk's function (F_8) with and without competition. Both runs consist of 512 individuals. For the competition they were distributed into four groups.

locate the attractor region of the global optimum much faster than runs without competition. The reason is that the group performing large steps is able to locate the attractor regions very fast. In the final stage of the search the group with the smallest steps locates the optimum with high precision.

A real challenge for any continuous function optimization program is to follow a steep curved valley which is only slightly decreasing. An example is the function of Rosenbrock [8]. In [10] the two dimensional function was extended to a n-dimensional function.

$$F_{16}(x) = \sum_{i=1}^{n} \left(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right) \quad -5.12 \leq x_i \leq 5.12 \tag{11}$$

For these kind of functions the BGA mutation scheme is not efficient. Line recombination seems much more promising. With line recombination first the direction is computed, then the mutation step. In Fig. 8 a competition is shown between discrete recombination and line recombination. One observes that first discrete recombination is exploring the search space. For a long time the population stays at the saddle point 1, then after the group using line recombination has taken over, it finally is able to leave the saddle point.

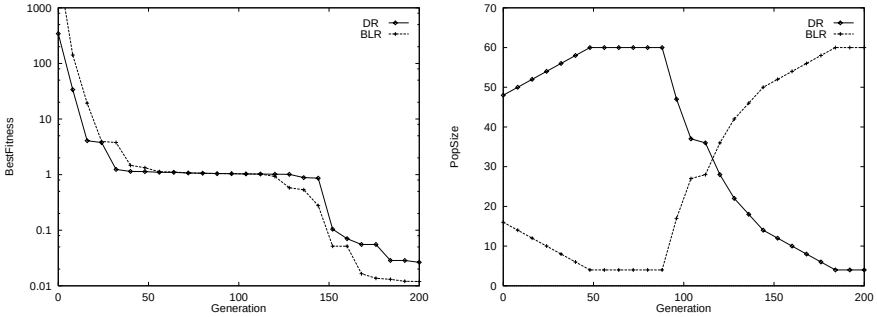

Fig. 8. Competition between discrete recombination (DR) and line recombination (BLR) for Rosenbrock's function of dimension 4. BLR takes over at generation 120.

6 Conclusion

Competition between subpopulations using different strategies makes the BGA search more effective and robust. The method presented in this paper is a first step. The strategies have to be defined at the start of the run, only their relative frequency is changed by competition. The user of the BGA has the responsibility to define a set of reasonable strategies for the given problem. The question to be investigated in the future is how robust the competition is in respect to the parameters used for the competition. All runs reported in this paper have been done with the same set of parameters.

References

1. Thomas Bäck. Self-Adaption in Genetic Algorithms. In Francisco Varela and Paul Bourgine, editors, *Towards a Practice of Autonomous Systems*, pages 263–271, 1992.
2. Thomas Bäck and Hans-Paul Schwefel. A Survey of Evolution Strategies. In *Proceedings of the Fourth International Conference of Genetic Algorithms*, pages 2–9, San Diego, 1991. ICGA.
3. Thomas Bäck and Hans-Paul Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1:1–24, 1993.
4. Terence C. Fogarty. Varying the probability of Mutation in the Genetic Algorithm. In J. David Schaffer, editor, *Proceedings of the Third International Conference of Genetic Algorithms*, pages 104–109. Morgan-Kaufman, 1989.
5. Michael Herdy. Reproductive Isolation as Strategy Parameter in Hierarchical Organized Evolution Strategies. In *PPSN 2 Bruxelles*, pages 207–217, September 1992.
6. Heinz Mühlenbein and Dirk Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm: Continuous Parameter Optimization. *Evolutionary Computation*, 1(1):25–49, 1993.
7. Heinz Mühlenbein and Dirk Schlierkamp-Voosen. The science of breeding and its application to the breeder genetic algorithm. *Evolutionary Computation*, 1(4):335–360, 1994.
8. H. H. Rosenbrock. An automatic method for finding the greatest or least value of a function. *The Computer Journal*, 3(3):175–184, 10 1960.
9. Fabio Schoen. A Wide Class of Test Functions for Global Optimization. *Journal of Global Optimization*, 3(2):133–137, 1993.
10. H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, 1981.