Predictive Models for the Breeder Genetic Algorithm

I. Continuous Parameter Optimization

Heinz Mühlenbein muehlen@gmd.de

Dirk Schlierkamp-Voosen dirk@gmd.de

GMD P.O. 1316
D-5205 Sankt Augustin 1, Germany

Abstract

In this paper a new genetic algorithm called the **Breeder Genetic Algorithm (BGA)** is introduced. The BGA is based on artificial selection similar to that used by human breeders. A predictive model for the BGA is presented which is derived from quantitative genetics. The model is used to predict the behavior of the BGA for simple test functions. Different mutation schemes are compared by computing the expected progress to the solution. The numerical performance of the BGA is demonstrated on a test suite of multimodal functions. The number of function evaluations needed to locate the optimum scales only as $n\ ln(n)$ where n is the number of parameters. Results up to $n = 1000$ are reported.

1 Introduction

In (Mühlenbein, Gorges-Schleuter & Krämer 1988), (Mühlenbein, 1991), (Mühlenbein, 1992b) the parallel genetic algorithm (PGA) was successfully applied to combinatorical problems. Continuous parameter optimization has been described in (Mühlenbein, Schomisch & Born, 1991). We have now substantially improved the results obtained with the PGA. The improvements are the result of a new genetic algorithm. This algorithm we call the **Distributed Breeder Genetic Algorithm (DBGA)**. The DBGA is inspired by breeding. Each one of a number of virtual breeders has the task of improving its own subpopulation. Occasionally a breeder

In: *Evolutionary Computation*, 1(1): 25-49, 1993.

gets a new individual from its neighboring breeders. A DBGA with a single virtual breeder is called a **Breeder Genetic Algorithm (BGA)**.

Whereas the PGA models natural and self-organized evolution, the DBGA is based on rational selection performed by human breeders. In biological terminology: the PGA models *natural selection* and the DBGA models *artificial selection*. We expected artificial selection to be more efficient for optimization than natural selection. We will show in this paper that this is indeed the case. This result does not imply that the DBGA is "better" than the PGA. Both algorithms are important, both from a biological and a computer science point of view. The DBGA models rational controlled evolution whereas the PGA models evolution which self-organizes.

The major goal of this paper is to show that the BGA can profit, both in theory and practice, from the long experience accumulated by human breeders. In the last two hundred years, starting just before Darwin, breeding of animals has advanced from an art based on intuition to an empirical science based on statistics. The BGA has been strictly designed according to this science of breeding. But until recently, there has been a major difference between a real breeder and the virtual breeder of our genetic algorithm. The human breeder does not have information about the genetic material, he has to estimate aggregrate values which he calls the *breeding value* of an animal. The virtual breeder of our GA has knowledge about all the genes of his population. Furthermore he controls the genetic operators (i.e. mutation, recombination etc.). But with the introduction of biotechnology this distinction will probably soon vanish!

The BGA is not radically new, it can be seen as a recombination between evolution strategies ES (Schwefel, 1981), (Bäck, Hoffmeister & Schwefel, 1991) and genetic algorithms GA (Goldberg, 1989). The BGA uses truncation selection as performed by breeders. This selection scheme is similar to the (μ,λ)-strategy in ES (Schwefel, 1981). The search process of the BGA is mainly driven by recombination, making the BGA a genetic algorithm. Mutation is an important background operator for the BGA. The mutation rate is inversely proportional to the number of parameters to be optimized and the mutation range is fixed.

The BGA is a random search method which can be applied to both discrete and continuous functions. In this paper the following questions will be answered

- Given a mutation scheme, what is the expected progress of successful mutations for a single individual?
- Given a selection and recombination schedule, what is the expected progress of the population?

This approach is opposite to the standard GA analysis, which starts with the schema theorem. Mutation and recombination are only considered as disruptions. We see mutation and recombination as constructive operators. They are evaluated according to the probability that they create better solutions.

The outline of the paper is as follows. In section 2 we formally describe the DBGA and the BGA. Mutation is analyzed in section 3. The approach of quan-

titative genetics to artificial selection is explained in section 4. The framework is used to investigate selection and recombination. The selection equation is used in the following section to analyze proportionate selection. It is shown in section 5 that proportionate selection is not a good scheme for optimization. The empirical laws derived in section 4 are investigated in section 6 for continuous parameter optimization. The interaction of recombination and mutation is analyzed in section 7. Numerical results for dimensions up to 1000 are reported in section 8.

We strongly believe that a good theory has to be proven with challenging applications. Therefore we only describe that part of the theory which is necessary for understanding the rationale of the BGA. The theory is explained in more detail in (Mühlenbein & Schlierkamp-Voosen, 1992a). This paper concentrates on the BGA. The DBGA is investigated in (Mühlenbein & Schlierkamp-Voosen, 1992b).

2 A formal description of the DBGA

The description is analogous to that in (Mühlenbein, Schomisch & Born, 1991) where the PGA is described. The DBGA is an eight tuple

$$DBGA = (P^0, sub, N, sg, \tau, BGA, F, term) \tag{1}$$

where

P^0 := initial population
sub := number of subgroups
N := number of individuals per subgroup
sg := number of neighboring subgroups
τ := migration schedule
BGA := Breeder Genetic Algorithm
F := fitness function
$term$:= termination criterion

Each subgroup is controlled by a BGA. A BGA can be described by

$$BGA = (P_g^0, N, T, ?, \Delta, HC, F, term) \tag{2}$$

P_g^0 is the initial subpopulation, N the size of the population, T the truncation threshold, $?$ the recombination operator, Δ the mutation operator and $term$ the termination criterion. HC is a hill-climbing method.

All numerical results in this paper have been obtained without local hill-climbing. We will show in the next section that the BGA mutation operator works almost as well as more specialized local hill-climbing methods which do not use derivatives of the function to be optimized. Therefore for continuous parameter optimization local hill-climbing is not as important as for discrete optimization problems. The importance of hill-climbing in discrete domains has been shown in (Mühlenbein, 1991), (Mühlenbein, 1992a).

We will now describe each operator in more detail. Their respective influence on the performance of the BGA will be investigated in the following sections. For a BGA run a set of genetic operators is defined. Then these operators are applied to the individuals of the population. There are no probabilities for the application of the operators.

2.1 Selection

The BGA uses a selection scheme called *truncation selection.* The $T\%$ best individuals are selected and *mated randomly* until the number of offspring is equal to the size of the population. The offspring generation replaces the parent population. The best individual found so far will remain in the population. Self-mating is prohibited.

2.2 Recombination

In (Mühlenbein, Schomisch & Born, 1991) we distinguished between recombination and crossing-over. The mixing of the variables was called recombination and the mixing of the values of a variable was named crossing-over. From a mathematical viewpoint this distinction is unnecessary. Any operator which combines the genetic material of two parents we will call a recombination operator in this paper. Recombination is the most important search operator of the BGA. We have implemented three different operators.

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

$$z_i = \{x_i\} \; or \; \{y_i\} \tag{3}$$

  x_i or y_i are chosen with probability 0.5.

- **Extended intermediate recombination**

$$z_i = x_i + \alpha_i(y_i - x_i) \quad i = 1, \ldots, n \tag{4}$$

  α_i is chosen uniform randomly in the interval $[-0.25, 1.25]$

- **Extended line recombination**

$$z_i = x_i + \alpha(y_i - x_i) \quad i = 1, \ldots, n \tag{5}$$

  α is chosen uniform randomly in $[-0.25, 1.25]$.

The geometric effect of recombination has been explained in (Mühlenbein, Schomisch & Born, 1991). Discrete recombination generates *corners* of the hypercube defined by the components of x and y. Extended intermediate recombination can generate

any point *within a slightly larger hypercube*. Extended line recombination generates a point on the *line* defined by x and y. In both operators the new point may lie outside $[x_i, y_i]$.

The rational of these operators are geometrically obvious. Discrete recombination and intermediate recombination with $\alpha = 0.5$ have been used successfully in genetic algorithms and evolution strategies. They have been already proposed by Bremermann (1966).

Each recombination operator has been designed to solve specific problems which have been encountered. In the course of applying the BGA to more problems we will extend the three basic operators in the future. The next recombination operator will be based on three-tuple mating. Here three points (parents) will be recombined. General m-tuple mating has already been suggested by Bremermann (1966).

2.3 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 $0.1 \cdot searchinterval_i$. $searchinterval_i$ is the domain of definition of variable x_i.

The new value z_i is computed according to

$$z_i = x_i \pm range_i \cdot \delta \tag{6}$$

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

$$\delta = \sum_{i=0}^{15} \alpha_i 2^{-i} \quad \alpha_i \in 0, 1$$

Before mutation we set $\alpha_i = 0$. Then each α_i is mutated to 1 with probability $p_\delta = 1/16$. 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

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

The mutation operator is similar in spirit to that used by the PGA (Mühlenbein, Schomisch & Born, 1991), but the BGA operator is much more easy to understand. Furthermore, it is independent of the location in phenotype space.

The standard BGA mutation operator is able to generate *any* point in the hypercube with center x defined by $x_i \pm range_i$. But it tests much more often in the neigborhood of x. In the above standard setting, the mutation operator is able to locate the optimal x_i up to a precision of $range_i \cdot 2^{-15}$. The rationale of this mutation operator will be explained in the next section.

3 Mutation in continuous domains

The mutation operator has been investigated for binary domains in (Mühlenbein, 1991), (Mühlenbein, 1992a). It was shown that the mutation rate should be inversely proportional to the number of bits in the chromosome. In this section we will show that an adaptation of the above strategy, the BGA mutation scheme, also works well in continuous domains.

Our analysis is similar to the investigation of mutation in evolution strategies (Bäck, Hoffmeister & Schwefel, 1991), (Schwefel, 1981), (Rechenberg, 1973) and in random search methods (Törn & Zilinskas, 1989), (Solis & Wets, 1981).

We will compare different mutation schemes according to the performance measure *expected progress E(r)*. Given an arbitrary point x with distance r to the optimum then $E(r)$ is defined as the expected improvement of x by successful mutations in euclidian distance. Mutations giving no improvement are not counted. $E(r)$ is defined by probability theory as an integral over the domain of successful mutations. The integrand is given by $progress(y) * probability(y)$. The domain of successful mutation depends on the fitness function. We will assume for the analysis a unimodal function with spherical symmetry.

First we will compute the expected progress in one dimension for different mutation schemes - uniform distributed mutation, normal distributed mutation and the BGA mutation scheme. We will show that the expected progress of the simple BGA scheme is only six times worse than normal distributed mutation with optimal adaptation.

Uniform distributed mutation

The mutation operator randomly chooses a number z in the interval defined by $[-A, A]$. A is called the *mutation range*. The new point is given by

$$x_m = x + z$$

Let $\|x\| = r$. For convenience let the optimum be in 0. Then for $A \geq 2r$ the expected progress of successful mutations is given by

$$E(r) = 2 \cdot \int_0^r \frac{x}{2A} dx = \frac{r^2}{2A} \tag{7}$$

The optimum progress is obtained for $A = 2r$. Similarly we have for $A \leq r$

$$E(r) = \int_0^A \frac{x}{2A} dx = \frac{A}{4} \tag{8}$$

The optimum progress is obtained for $A = r$. The formulas for the case $r \leq A \leq 2r$ are slightly more difficult and omitted here. This proves the next theorem.

Theorem 1 *For uniform distributed mutation the optimal mutation range is given by A=2r or A=r. In both cases the normalized expected progress is given by*

$$\frac{E(r)}{r} = \frac{1}{4} \tag{9}$$

Remark: If the mutation range is held constant, then the normalized expected progress goes to zero for $r \longrightarrow 0$.

We will now investigate a distribution which more often chooses points in the neighborhood of r.

Normal distributed mutation

Normal distributed mutation is used in evolution strategies (Bäck, Hoffmeister, & Schwefel, 1991). The mutation operator chooses a random number z from a normal distribution $N(0,\sigma)$ with standard deviation σ.

z will be a successful mutation if $-2r < z < 0$. The expected progress for these mutations can be computed as usual

$$E(r,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\left(\int_{-r}^{0} -x \exp\frac{-x^2}{2\sigma^2}dx + \int_{-2r}^{-r}(2r+x)\exp\frac{-x^2}{2\sigma^2}dx\right) \tag{10}$$

After tedious, but straightforward integration one obtains

$$E(r,\sigma) = \frac{\sigma}{\sqrt{2\pi}}\left(1 - 2\exp\frac{-r^2}{2\sigma^2} + \exp\frac{-2r^2}{\sigma^2}\right) + \sqrt{\frac{2}{\pi}}r\int_{-2\frac{r}{\sigma}}^{-\frac{r}{\sigma}} \exp -\frac{x^2}{2}dx \tag{11}$$

The standard deviation giving the optimal expected progress can be obtained from $\frac{dE}{d\sigma} = 0$. We will not compute it here, because for high dimensions ($n \gg 0$) Rechenberg(1973) was able to compute it approximately. We will use Rechenberg's result later when we investigate n dimensions. From the above equation we obtain by Taylor series expansion

Theorem 2 *For fixed σ and $r \longrightarrow 0$ the expected progress is approximate*

$$E(r,\sigma) \approx \frac{r^2}{\sqrt{2\pi}\sigma} \tag{12}$$

For $\sigma = r$ one obtains the normalized progress

$$\frac{E(r,\sigma)}{r} \approx 0.1 \tag{13}$$

Remark: If σ is held fixed then the normalized expected progress goes to zero for $r \longrightarrow 0$. A constant normalized expected progress is obtained if σ is proportional to r.

Both, uniform and normal distributed mutation need an adaptation of the mutation range in order to give a reasonable progress for small r. We will now show that the BGA mutation scheme does not need such an adaptation.

The BGA mutation scheme

The BGA mutation operator randomly chooses with probability 1/32 one of the 32 points

$$\pm\left(2^{-15}A, 2^{-14}A, \ldots, 2^{0}A\right) \tag{14}$$

A defines the mutation range. The BGA mutation scheme can approximate the optimum only up to a precision of $2^{-15}A$ [4712].

Theorem 3 *The normalized expected progress for the BGA mutation scheme is bounded by*

$$\frac{1}{32} \leq \frac{E(r)}{r} \leq \frac{1}{16} \tag{15}$$

Proof: For simplicity, let for some $0 \leq k \leq 15$

$$A = 2^k r$$

Successful mutations have to be in the range $-2r < z < 0$. Then the expected progress is given by

$$E(r) = \frac{1}{32} r \cdot (1 + \cdots + 2^{-15+k}) \tag{16}$$

■

The normalized expected progress for the BGA mutation scheme is of the same order as the optimal schemes for uniform or normal distributed mutation. But the optimal schemes need the exact distance r to the optimum. In a real application the distance is not known. Therefore the mutation range has to be empirically adapted (Bäck, Hoffmeister & Schwefel, 1991).

The BGA mutation is robust. It does not need an adaptation of the mutation range because it generates points more often in the neighborhood of the given point. Nevertheless the BGA mutation is only about three times less efficient than normal distributed mutation with $\sigma = r$.

We will now investigate mutation in high dimensions. We will start with the BGA mutation because it is the simplest to analyze.

The BGA mutation in n dimensions

A variable is chosen with probability $p_m = 1/n$ for mutation. Therefore on average just one variable will be mutated.

Theorem 4 *Given a point with distance $r \leq A$ to the optimum, then the expected progress in n dimensions is bounded by*

$$\frac{1}{32n} \leq \frac{E(n,r)}{r} \leq \frac{1}{16n} \tag{17}$$

Proof Let an arbitrary point with distance r be given. By rotation we move the point to the first axis. This will give the point $(r, 0, \ldots, 0)$. The rotation does not change the average progress because we asumed that the fitness function is spherical symmetric. Therefore

[4712] If a higher precision is needed, the mutation range will be reduced during the run. We will discuss this modification at the end of the section.

$$E(n,r) = \frac{1}{n} E_{x_1}(r)$$

$E_{x_1}(r)$ is the average progress in the direction of x_1. The result now follows from theorem 3. ∎

The normalized expected progress decreases with $1/n$. We will now consider uniform distributed mutation.

Uniform distributed mutation in n dimensions

The mutation operator chooses n-times a number from the interval defined by $[-A, A]$. Let a point with distance r to the optimum be given.

Theorem 5 *The expected progress for uniform mutation is given for $A \geq 2r$ by*

$$\frac{E(n,r)}{r} = \frac{\pi}{2n(n+1)} \frac{r^{n+1}}{A^n} \tag{18}$$

The optimal expected progress is obtained for $A = 2r$

$$\frac{E(n,r)}{r} = \frac{\pi}{2n(n+1)} \cdot 2^{-n} \tag{19}$$

Proof: We start with $n = 2$. Let $\|x\| = r_1$. The expected progress can be obtained from the integral

$$E(2,r_1) = \frac{1}{4A^2} \int_0^{r_1} \int_0^{2\pi} (r_1 - r) r d\phi dr$$

Therefore

$$E(2,r_1) = \frac{\pi}{12} \frac{r_1^3}{A^2}$$

For arbitrary n the expected progress is given by

$$E(n,r_1) = \frac{1}{(2A)^n} \int_0^{r_1} \int (r_1 - r) dh_n(r) dr$$

The inner integral has to be done in polar coordinates over the surface of the n-dimensional hypersphere of radius r. We computed

$$E(n,r_1) = \frac{\pi}{2n(n+1)} \frac{r_1^{n+1}}{A^n} \tag{20}$$

From this equation the theorem immediately follows. ∎

The optimal normalized average progress for uniform distributed mutation decreases exponential with n. The reason for this behavior is the well known fact that the

volume of the unit sphere in n-dimensions goes to zero for $n \longrightarrow \infty$. Better results can be obtained if the uniform distribution is restricted to a hypersphere with radius r_h. For $r_h = 1.225r/\sqrt{n}$ the expected progress was computed for large n and small r in (Schumer & Steiglitz, 1968)

$$\frac{E(n,r)}{r} = \frac{0.2}{n} \tag{21}$$

We now turn to normal distributed mutation.

Normal distributed mutation in n dimensions

The mutation operator chooses n-times a number randomly from a normal distribution $N(0,\sigma)$ with standard deviation σ. Let $\|x\| = r_1$.

The expected progress is given by

$$E(r_1,\sigma) = \frac{1}{(\sqrt{2\pi}\sigma)^n}\left(\int_{-r_1}^{0}\int -r\exp -\frac{r^2}{2\sigma^2}dh_n dr + \int_{-2r_1}^{-r1}\int (2r_1+r)\exp -\frac{r^2}{2\sigma^2}dh_n dr\right) \tag{22}$$

The inner integral has to be done in polar coordinates over half of the surface of the n-dimensional hypersphere of radius r. This integral does not appear to be intractable. Nevertheless, we did not find a closed solution. However Rechenberg was able to compute the expected progress approximately for $n \gg 0$. We just cite the result (Rechenberg, 1973).

Theorem 6 *(Rechenberg): The optimal standard deviation for normal distributed mutation is given by*

$$\sigma_{opt} = 1.2\ \frac{r}{n}$$

The expected progress is approximately

$$\frac{E(n,r,\sigma)}{r} = \frac{0.2}{n} \tag{23}$$

We now summarize the results of the above analysis. The results have been proved for unimodal fitness functions with rotational symmetry only.

The expected progress for the simple BGA scheme is $4-6$ times less than the expected progress for the normal distributed mutation with optimal σ. It scales in n dimensions like $1/n$ which is believed to be the optimal progress rate for search methods which do not use derivatives of the function. We may thus propose the following rule for high dimensions.

Rule 1 *Do not change too many variables at the same time in high dimensions*

The above rule was already found by Bremermann (1966). For problems of the kind

$$\|e\| = \|b - Mx\| \quad minimal$$

he suggested uniform mutation with a mutation range of

$$A = \frac{\|e\|}{n}$$

This is a very simple and *constructive* adaptation scheme. But unfortunately an analysis immediately shows that the normalized expected progress gets worse with increasing $\|M\|$.

The formulas of this section can be used to estimate the number of trials to approximate the optimum with a given precision. For simplicity, let us assume a population of size 1. In this case the BGA accepts a mutated string only if it is not worse than the parent string.

Theorem 7 *For simplicity let the optimum be at $x = 0$. Let the expected progress be of the form $E(\|x\|) = c\|x\|/n$. Then the number of iterations IT needed to approximate the optimum up to a precision of ϵ is given for $n \gg 0$ by*

$$IT = \frac{n}{c} ln \frac{\|x_0\|}{\epsilon} \tag{24}$$

Proof: Let x_0 be the initial string, r_0 its euclidian norm. In a very crude statistical approximation we have

$$\|x_{i+1}\| = \|x_i\|(1 - \frac{c}{n})$$

Therefore we get

$$\|x_i\| = \|x_0\|(1 - \frac{c}{n})^i \tag{25}$$

The number of iterations IT can be computed from the equation

$$i \cdot ln(1 - \frac{c}{n}) = ln(\frac{\|x_i\|}{\|x_0\|})$$

by Taylor series expension and $\|x_{IT}\| = \epsilon$. ∎

Remark: The number of iterations linearly increases with n, if the initial string has the same euclidian distance in all dimensions.

The above theorem was derived without statistical rigour. Therefore we investigated the mutation scheme with simulations. In table 1 numerical results are given for the initial string $x_0 = (1, 1, \ldots, 1)$. Note that the euclidian distance of this string is $\sqrt{n}$. Therefore the number of iterations should approximately increase like $nln(n)$. The fitness function is the euclidean distance.

The table 1 confirms the statistical analysis. The number of iterations increases like $32n \cdot ln(10^3\|x_0\|)$. We would like to mention that Solis and Wets(1981) report a

n	p_m	IT	SD	$32n \cdot ln(10^3\sqrt{n})$
10	0.1	2240	424	2578
10	0.2	2197	433	
10	0.4	6090	1177	
20	0.05	4855	701	5380
20	0.1	5296	780	
20	0.2	12131	1390	
100	0.01	33318	2101	29473
100	0.02	33772	1637	
100	0.04	83150	5741	

Table 1: Number of iterations IT, termination $\epsilon = 10^{-3}$

constant of 33 for their random search method. Their method dynamically adjusts the range of mutation.

The number of iterations changes dramatically if the mutation rate grows. A very small change (in absolute terms) of the mutation rate from 1/100 to 4/100 has a huge impact. We have explained this behavior for discrete functions in (Mühlenbein,1992a).

The BGA mutation scheme can still be improved by two or more discrete adaptation steps. The adaptation works if not just a single point but a population of points are used. We outline the idea with an example. We restrict the BGA mutation range to say 9 points instead of 15

$$(2^{-8}A, \ldots, 2^0 A)$$

Then if all points of the population are within say a range of $2^{-7} \cdot A$, we change the mutation range to

$$A' = 2^{-7} \cdot A$$

This procedure is equivalent to dynamic parameter encoding techniques (Schraudolph & Belew, 1992). But note that the above procedure reduces the robustness of the mutation scheme. After a discrete adaptation step, points outside the new range cannot be generated by mutation anymore.

4 The response to selection

In this section we investigate the expected progress to the solution for the discrete recombination operator. Compared to mutation the following difficulty arises. Recombination cannot be analyzed for a single point (individual), it needs at least two points. Furthermore recombination is not a random process, it depends on the

Figure 1: Regression of truncation selection

Breeders often use *truncation selection* or *mass selection* as shown in figure 1. In truncation selection with threshold T, the $T\%$ best individuals will be selected as parents. T is normally chosen in the range 50% to 10%.

The prediction of the response to selection starts with

$$R(t+1) = b_t \cdot S(t) \tag{28}$$

The breeder either measures b_t in previous generations or estimates b_t by different methods (Crow, 1986). It is normally assumed that b_t is constant for a certain number of generations. This leads to

$$R(t+1) = b \cdot S(t) \tag{29}$$

There is no genetics involved in this equation. It is simply an extrapolation from direct observation. The prediction of just one generation is only half the story. The breeder (and the GA user) would like to predict the cumulative response R_n for n generations of his breeding scheme.

$$R_n = \sum_{t=1}^{n} R(t) \tag{30}$$

In order to compute R_n a second equation is needed. In quantitative genetics, several approximate equations for $S(t)$ are proposed (Bulmer, 1980), (Falconer, 1981). Unfortunately these equations are not useful in the case of haploid organisms as used in our BGA. Therefore, we can only apply the research methods of quantitative genetics, not the results.

Our approach has been influenced by Robertson, who wrote in (Robertson, 1970): "We may by conventional analysis discover that factor A and factor B have significant effects and that there is a significant interaction between them. It is however much more useful to find that an analysis in terms of, say, A/B accounts for almost all the variations due to both factors. In statistical terms, we are seeking the best 're-parameterization'". We will show that a re-parameterization is possible for the BGA.

If the fitness values are normally distributed, the selection differential $S(t)$ in truncation selection is approximately given by

$$S = I \cdot \sigma_p \tag{31}$$

where σ_p is the standard deviation. I is called the *selection intensity*. The formula is a feature of the normal distribution. A derivation can be found in (Bulmer, 1980). In table 2 the relation between the truncation threshold T and the selection intensity I is shown. A decrease from 50% to 1% leads to an increase of the selection intensity from 0.8 to 2.66.
If we insert (31) into (29) we obtain the well-known *selection equation* (Falconer, 1981)

$$R(t+1) = b \cdot I \cdot \sigma_p(t) \tag{32}$$

The science of artificial selection consists of estimating b and $\sigma_p(t)$. The estimates depend on the fitness function. We will use as an introductory example the binary

T	80 %	50 %	40 %	20 %	10 %	1 %
I	0.34	0.8	0.97	1.2	1.76	2.66

Table 2: Selection intensity.

$ONEMAX$ function. Here the fitness is given by the number of 1's in the binary string.

In principle, the selection equation can be used for any populations. But the quality of the prediction depends on the size of the population and the genetic operators. The response to selection for the mutation operator is erratic for small populations. Therefore we have used in section 3 probability theory to predict the outcome of *many* trials. The behavior of recombination is much more predictable. For this reason we consider this case first. We assume *uniform crossing-over* for recombination (Syswerda, 1989). Uniform crossing-over is similar to discrete recombination in continuous domains.

We will first estimate b. A popular method for estimation is to make a regression of the midparent fitness value to the offspring. The midparent fitness value is defined as the average of the fitness of the two parents. A simple calculation shows that the probability of the offspring being better than the midparent is equal to the probability of them being worse. Therefore the average fitness of the offspring will be the same as the average of the midparents. The result means that the average of the offspring is the same as the average of the selected parents. This gives $b = 1$ for $ONEMAX$.

Estimating $\sigma_p(t)$ is more difficult. We make the assumption that uniform crossing-over is a random process which creates a binomial fitness distribution with probability $p(t)$. $p(t)$ is the probability that there is a 1 at a locus. Therefore the standard deviation is given by

$$\sigma_p(t) = \sqrt{n \cdot p(t) \cdot (1 - p(t))} \tag{33}$$

Noting that $M(t) = n \cdot p(t)$ we obtain from (26) and (32) the difference equation

$$p(t+1) - p(t) = \frac{I}{\sqrt{n}} \cdot \sqrt{p(t) \cdot (1 - p(t))} \tag{34}$$

The difference equation can be approximated by a differential equation

$$\frac{dp(t)}{dt} = \frac{I}{\sqrt{n}} \cdot \sqrt{p(t) \cdot (1 - p(t))} \tag{35}$$

The initial condition is $p(0) = p_0$. The solution of the differential equation is given by

$$p(t) = 0.5 \left(1 + \sin(\frac{I}{\sqrt{n}} t + \arcsin(2p_0 - 1)) \right) \tag{36}$$

n	I	N	FIT	GEN	FE
64	0.8	24	59.4	22.2	532.8
	1.2	32	61.4	13.4	428.8
	1.6	48	59.5	9.5	456.0
128	0.8	32	117.0	29.5	944.0
	1.2	48	121.5	18.9	907.0
	1.6	64	119.5	13.9	889.6
256	0.8	64	247.0	44.1	2822.4
	1.2	96	252.0	27.6	2649.6
	1.6	128	245.2	20.6	2636.8
512	0.8	128	506.2	61.8	7910.4
	1.2	192	507.7	39.3	7545.6
	1.6	256	504.8	30.5	7808.0
512	0.8	256	511.8	57.6	14745.6
	1.2	384	512.0	36.6	14054.4
	1.6	512	511.6	28.6	14643.2

Table 3: Averages over 50 runs for ONEMAX

The convergence of the total population is characterized by $p(GEN) = 1$. GEN can be easily computed from the above equation It is given by

$$GEN = (\frac{\pi}{2} - \arcsin(2p_0 - 1)) \cdot \frac{\sqrt{n}}{I} \tag{37}$$

We have now arrived at a very interesting result. The number of generations needed until convergence is proportional to $\sqrt{n}$ and inversely proportional to $1/I$.

Because this result was obtained using a number of assumptions, we also investigated the problem with extensive simulations. Some numerical results for the breeder genetic algorithm BGA with uniform crossing-over are given in table 3. Mutation is not used.

We will try to capture the main results of table 3 in three empirical laws.

Empirical Law 1 *If the size of the population N is large enough that the population will converge to the optimum and the initial population is randomly generated ($p(0) = 0.5$), then the number of generations needed to converge to the optimum is given by*

$$GEN = k \cdot \frac{\sqrt{n}}{I} \qquad 0.8 \leq I \leq 1.6 \quad k \approx 2 \tag{38}$$

Note that GEN is only slightly larger than the analytically computed solution. GEN is independent of N for large N [4713]. It is a common belief by breeders

[4713]The minimum N for which the population will converge seems to be proportional to $\ln(n)\sqrt{(n)} \cdot f(I)$ for $I \geq 0.5$.

I	0.2	0.35	0.5	0.8	1.0	1.2	1.7	2.1
MinPop	90	64	50	48	54	65	90	200
GEN	79	44	32	21	15.5	13.5	10.5	8.3
FE	7110	2816	1600	1008	837	877	945	1660

Table 4: Minimal population size MinPop, ONEMAX, $n = 64$

(Falconer, 81) and GA researchers, that GEN depends mainly on N. But our simulations show that this is not the case, at least for the ONEMAX function. GEN only depends on the size of the problem n and the selection intensity I. Furthermore, increasing I by a factor of two halves the number of generations required. This result seems to suggest that a high selection intensity could be the best selection scheme. But this is of course not true. A high selection intensity leads to premature convergence and a bad quality of solution.

In table 4 we investigate the minimal population size $MinPop(I)$ for which the population will converge with given probability to the optimum. It is easily verified that $MinPop$ has to increase for very large I. In this case just the best individual is selected. Then uniform crossing-over will only duplicate this individual. The population converges in one generation. Therefore the optimal solution has to be contained in the initial population. From standard probability theory we obtain $MinPop \geq O(2^n)$.

The same argument shows that $MinPop$ increases for very small selection intensities. Consequently there has to be a selection intensity for which $MinPop$ is minimal. In table 4 $MinPop$ is defined by the condition that 60% of the runs should terminate with the optimal solution. The smallest $MinPop$ is obtained for $I = 0.8$. The best efficiency in function evaluations FE is at $I = 1$. The efficiency increases only slightly between $I = 1.0$ and $I = 1.6$. Therefore we normally run the BGA with a selection intensity of $I = 1.2$. Note that these results have been derived for uniform crossing-over and the ONEMAX function only.

The next empirical law can be derived from tables 3 and 4.

Empirical Law 2 *The number of generations until convergence is inversely proportional to I, if the same cumulative gain R_{GEN} is required*

$$GEN \propto 1/I \quad 0.8 \leq I \leq 1.6 \tag{39}$$

The third empirical law is a very crude estimate only.

Empirical Law 3 *The cumulative gain (i.e the total response to selection) is a monotonic function G of N/I for* $0.8 \leq I \leq 1.7$

$$R_\infty \approx G(N/I) \quad 0.8 \leq I \leq 1.7 \tag{40}$$

From the third law we obtain as a rule of thumb

Corollary 1 *The minimal amount of computation in function evaluations* $FE = N \cdot GEN$ *to get the same total response is independent of the selection intensity.*

Proof

$$N/I \approx G^{-1}(R_\infty)$$

If we assume $GEN = k/I$ for some k the result follows. ∎

We summarize the results. Uniform crossing-over and truncation selection can be analyzed with the methods developed in quantitative genetics. Thresholds in the range $50\% \leq T \leq 10\%$ give good results in terms of efficiency.

Before we investigate truncation selection and recombination in the continuous domain, we will show that the above framework can also be applied to proportionate selection.

5 Natural (proportionate) selection

Proportionate selection is used by the simple genetic algorithm (Goldberg, 1989). Let $0 \leq p_i \leq 1$ be the proportion of genotype i in a population of size N, F_i its fitness. Then the average fitnesss of the population is defined by

$$M(t) = \sum_{i=1}^{N} p_i(t) F_i \tag{41}$$

In proportionate selection the phenotype distribution of the selected parents is given by

$$p_{i,S} = \frac{p_i(t) F_i}{M(t)} \tag{42}$$

Theorem 8 *In proportionate selection the selection differential is given by*

$$S(t) = \frac{\sigma_p^2(t)}{M(t)} \tag{43}$$

Proof

$$\begin{aligned} S(t) &= \sum_{i=1}^{N} p_{i,S} F_i - M(t) \\ &= \sum_{i=1}^{N} \frac{p_i(t) F_i^2 - p_i(t) M^2(t)}{M(t)} \\ &= \frac{1}{M(t)} \cdot \sum_{i=1}^{n} p_i(t)(F_i - M(t))^2 \end{aligned}$$

■

We can compare truncation selection with proportionate selection by rewriting equation(43)

$$S(t) = \frac{\sigma_p(t)}{M(t)} \cdot \sigma_p(t) \tag{44}$$

Equation (44) shows that the selection intensity in proportionate selection *decreases* with the inverse of the average fitness and proportionately to the standard deviation. The closer the population comes to the optimum, the less severe is the selection! *Proportionate selection is afraid of reaching the goal.* This result explains why proportionate selection is not a good strategy for optimization purposes (DeJong, 1992). Many application oriented genetic algorithms use modifications of the proportionate selection scheme. Our analysis has shown that these modifications are necessary, not tricks to speed up the algorithm.

A recent overview and analysis of different selection schemes can be found in (Goldberg & Deb, 1991). Goldberg uses as performance measure the takeover time. Takeover is defined as the number of generations needed for convergence if the *optimum* is already *contained* in the population. We suggest analysis of the different selection schemes along the lines presented in this section.

6 Discrete recombination

In this section we will show that the major results of section 4 are also approximately valid for continuous functions with a unimodal macro structure. We take as representative examples the unimodal function

$$F_0(x) = \sum_{i=1}^{n} |x_i| \tag{45}$$

and the multimodal Rastrigin function

$$F_6(x) = n\ A + \sum_{1}^{n} \left(x_i^2 - A\ cos\,(2\pi x_i)\right) \qquad -5.12 \leq x_i \leq 5.12, A = 10.0 \tag{46}$$

F_6 has a large number of local minima, but they are unimodally distributed. On a large scale the structure of F_6 is like a quadratic function. The best minima are centered around 0.

The simulations have been done with discrete recombination and truncation selection. Discrete recombination for continuous functions is similar to uniform crossover in binary functions. In this case the BGA tries to solve a discrete problem with N alleles at each locus. Intermediate recombination as described in section 2 is more probabilistical. It can generate new alleles which are not in the initial population. In this case the results of the discrete case cannot be applied.

N	F	I	GEN	SD	FE	DF
256	F_0	1.6	13.9	1.6	3801	7.56
256	F_6	1.6	13.8	1.0	3788	29.12
192	F_0	1.2	18.2	2.8	3686	6.55
192	F_6	1.2	18.2	1.4	3798	25.06
128	F_0	0.8	27.3	3.9	3622	8.40
128	F_6	0.8	28.4	3.5	3916	29.82

Table 5: BGA with discrete recombination, n=20, 20 runs

Table 5 shows the simulation results for different selection intensities I and population sizes N. DF is the distance of the best value found to the global minimum of the function. The number of variables is 20.

GEN denotes the number of generations until convergence, FE the amount of computation in function evaluations ($FE = N \cdot GEN$). Convergence means that all individuals have the same value. SD is the standard deviation of GEN. The simulation results are in accordance with *empirical law 2*. The quality of the solution depends on N/I. The number of generations needed until convergence is inversely proportional to I. Therefore the amount of computation is almost the same in all cases.

The difference between the results for F_0 and the multimodal function F_6 is surprisingly small. The number of generations GEN until convergence is the same for both problems. For F_0 a better quality of approximation is obtained. This result shows the advantage of truncation selection, which does not take into account smaller differences of the fitness functions.

Table 6 shows the dependence of the quality of the solution on the size of the population. The selection intensity is fixed. The number of generations GEN until convergence increases approximately logarithmically with the size of the population N. At first this seems to contradict *empirical law 1*. But a closer look shows that the assumptions of the law are not fulfilled in continuous domains. The size of the population , necessary to reach the optimum with discrete recombination only, has to be infinite. But GEN increases much slower than N, the usual premise in quantitative genetics (Falconer, 1981).

In table 7 simulation results for different numbers of variables are given. The results are in accordance to *empirical law 2*. But also *empirical law 1* is approximately valid. In the column $QUOT$ the quotient $GEN(2n)/GEN(n)$ is computed. *Empirical law 1* predicts $QUOT$ to be $\sqrt{2}$. The computed $QUOT$ in table 7 is less than the predicted value. The reason for this behavior is similar to that before. If N is held fixed, then the quality of the solution obtained gets worse with higher dimensions. This means that the population will converge faster than a larger population which would give the same quality.

The next figures show the dynamic behavior of truncation selection. In figure 2

N	GEN	SD	DF	FE
32	9.7	2.2	127.0	342
128	15.8	2.9	42.9	2150
256	19.9	3.5	20.6	5350
512	23.4	2.5	10.1	12492
1024	25.6	2.3	5.4	27238

Table 6: Quality of the solution for F_6, I=1.2, n=20

n	N	I	GEN	SD	DF	QUOT
20	1024	1.6	18.5	1.8	8.4	
40	1024	1.6	24.5	2.5	27.4	1.32
80	1024	1.6	31.8	4.7	79.8	1.30
160	1024	1.6	42.7	8.5	239.3	1.34
20	768	1.2	24.0	2.0	6.9	
40	768	1.2	33.3	3.1	22.1	1.39
80	768	1.2	44.8	5.4	65.5	1.35
160	768	1.2	58.6	6.5	182.4	1.31
20	512	0.8	39.8	2.9	9.1	
40	512	0.8	50.7	5.0	26.4	1.27
80	512	0.8	65.0	9.7	77.2	1.28
160	512	0.8	87.7	14.4	222.6	1.35

Table 7: Quality of the solution for F_6 for constant N/I

$R(t)$ is shown for three selection intensities I and three population sizes N. N is chosen from table 5, so that approximately the same quality of solution is obtained. The $R(t)$ curves behave as expected. A strong selection intensity I in connection with a large population gives a much higher $R(t)$ at the beginning. Then it rapidly falls to zero.

In figure 3 the progress of the best solution found is shown. We see that the three different runs behave very similarly. For the same number of function evaluations all three simulation runs give the same best fitness. This is a stronger result than that of table 5, where only the amount of computation at convergence is reported.

In figure 4 the quotient $R(t+1)/S(t)$ is shown. We see that it oscillates around 1. The largest population oscillates the least. The regression coefficient b is 1 for all runs. This result can be explained in a similar way to the discrete case in section 4.

Figure 5 shows the fitness distribution of the population at generations $0, 2, 4, 6$. The fitness is symmetric about the average. It resembles a truncated normal distribution.

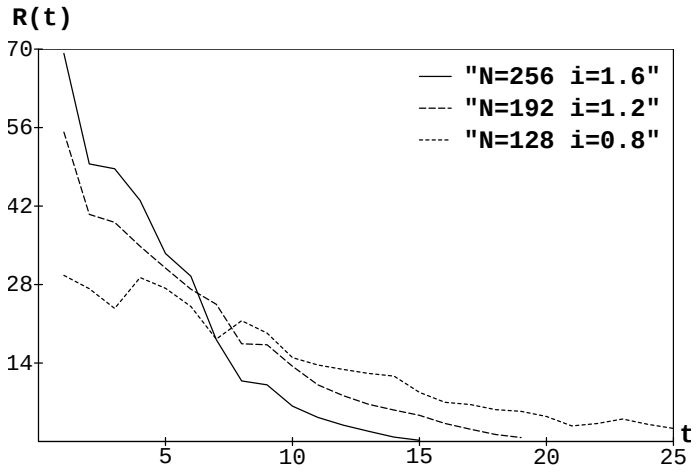

Figure 2: Response to selection (in generations. t)

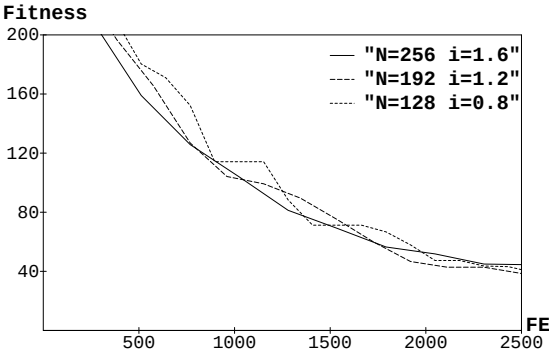

Figure 3: Fitness of the best individual

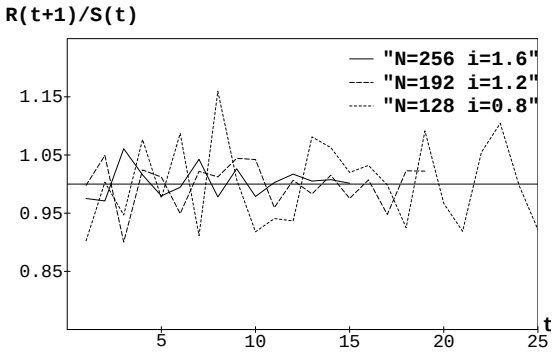

Figure 4: $R(t+1)/S(t)$

7 Mutation and recombination

In this section we will compare a BGA with mutation, with recombination and with mutation and recombination together. The search strategies of mutation and recombination are very different. Mutation is based on chance. It works in small populations most efficiently. For unimodal fitness functions the optimal size of the population is 1. The progress for a single mutation step is almost unpredictable. It needs probability theory and many trials to predict the behavior of this search operator. This analysis was done in section 3.

Recombination is based on restricted chance. The bias is given by the current population. Discrete recombination only shuffles the alleles contained in the population. The alleles of the optimum have to be present in the initial population. Otherwise recombination is not able to locate the optimum. The outcome of recombination is predictable by the selection equation if the population size is large.

Table 8 compares the BGA with discrete recombination, intermediate recombination, with mutation and with both mutation and recombination.

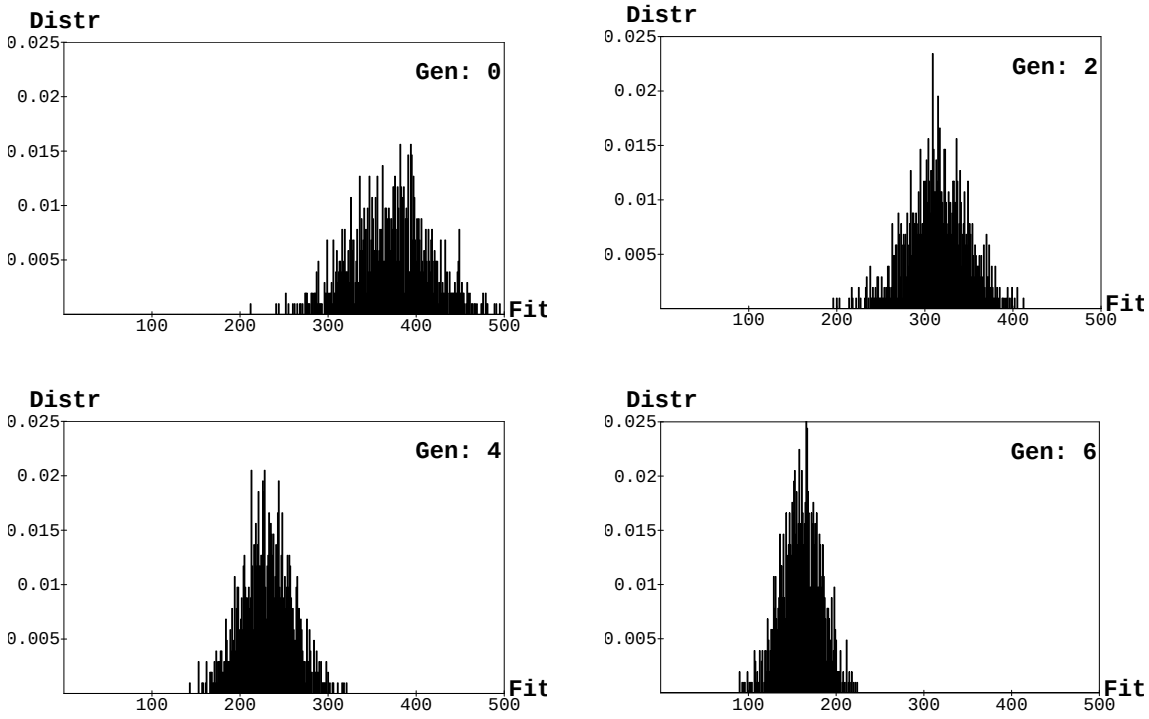

Figure 5: Fitness distribution of F6, n=20, N=1024

Table 8 contains the most important result of this paper.

A BGA with recombination and mutation outperforms a BGA with a single genetic operator. Mutation alone is more efficient than recombination alone. Mutation and recombination together have a synergetic effect.

Intermediate recombination behaves similarly to discrete recombination. It is a good search strategy as long as the population does not get to similar.

We will next show that in principle the selection equation can also be used for a BGA with mutation and recombination. In a large population recombination will be the dominating factor until the individuals get similar. The response to selection is predictable. In a small population mutation will dominate. The response to selection will be soon unpredictable. This behavior can be observed in figure 6. The response to selection R curve is smooth for the large population. It oscillates for the small population. The same behavior can be observed for the regression coeffient b in figure 7. For the large population b is approximately 1 until generation 7. This value was predicted for discrete recombination. From generation 20 on the coefficient erraticly behaves for both populations.

N	OP	GEN	SD	DF	FE
1	M	8132.0	1918.0	0.00007	8132
8	DR&M	855.0	176.0	0.00007	6850
20	DR	5.3	2.0	196.60000	126
20	IR	50.0		98.70000	1000
20	M	564.3	51.2	0.00009	11306
20	IR&M	420.0	51.2	0.00009	8420
20	DR&M	341.6	78.7	0.00009	6852
256	DR	19.9	3.5	20.60000	5350
256	M	252.1	10.7	0.00009	64793
256	DR&M	89.0	4.1	0.00009	23040

Table 8: Recombination(discrete DR, intermediate IR), mutation M for F_6

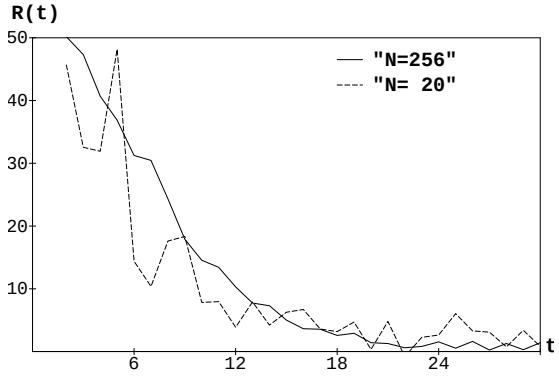

Figure 6: Response to Selection.

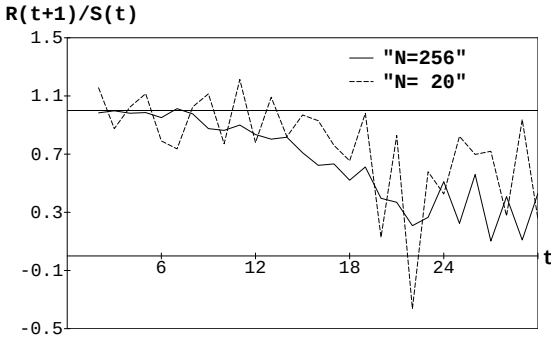

Figure 7: R(t+1)/S(t)

8 Numerical results

The efficiency of the standard BGA will be demonstrated with a set of test functions. An extensive performance evaluation for continuous function optimization has been done for the PGA in (Mühlenbein, Schomisch & Born, 1991). It was shown in that analysis that varying the problem size n gives valuable information about the efficiency of the search method. Efficiency is defined in number of function evaluations needed to solve the problem. We will show in this section how valuable this information is for a comparison.

We will present results for the test functions shown in table 11. Functions F_6-F_8 are described in (Mühlenbein, Schomisch & Born, 1991). They are often used as test functions in global optimization (Törn & Zilinskas, 1989). Function F_9 has been proposed by Ackley (1987). It has been subsequently used by Schwefel et al. for comparisons of different evolutionary algorithms (submitted for publication).

Our results are shown in tables 9 and 10. The functions F_6, F_8 and F_9 have

Rastrigin's Function F_6				Schwefel's Function F_7			
n	N	FE	$49n \cdot ln(n)$	n	N	FE	$292n \cdot ln(n)$
20	20	3608	2935	20	500	16100	17495
100	20	25040	22565	100	1000	92000	134471
200	20	52948	51923	200	2000	248000	309422
400	20	112634	117433	400	4000	700000	699803
1000	20	337570	338480				

Table 9: Termination criterion $\epsilon = 10^{-1}$ for F_6, $\epsilon = 10^{-4}$ for F_7.

Griewangk's Function F_8				Ackley's Function F_9			
n	N	FE	$668n \cdot ln(n)$	n	N	FE	$79n \cdot ln(n)$
20	500	66000	40023	30	20	19420	4733
100	500	361722	307625	100	20	53860	36380
200	500	748300	707855	200	20	107800	83713
400	500	1630000	1600920	400	20	220820	189330
				1000	20	548306	545713

Table 10: Termination criterion $\epsilon = 10^{-3}$ for F_8, $\epsilon = 10^{-3}$ for F_9.

been solved with a constant population size. These functions have a unimodal distribution of the local minima. The BGA mutation scheme is therefore able to find the minimum.

A different behavior is to be expected for Schwefel's function F_7. F_7 does not have a unimodal macrostructure. The best minima are far away from each other. Furthermore, the global minimum is located at the boundary of the search domain (Mühlenbein, Schomisch & Born, 1991). For F_8 extended intermediate recombination was used.

In the tables 9 and 10 the termination criterion term is fulfilled if one of the objectives $|F^{BGA} - F^{best}| \leq \epsilon \cdot |F^{best}|$ or $|F^{BGA} - F^{best}| \leq \epsilon$ if $F^{best} = 0$ is achieved.

The search time in function evaluations scales almost exactly with $n \cdot ln(n)$ for the functions F_6 and F_8. It scales linearly for function F_9 in the range $n = 100$ till $n = 1000$. Only function F_7 gives slightly different results.

These results can be predicted by the BGA theory. All the test functions have a global structure which makes them easy to optimize. The results of the BGA are better than those of the PGA. The function evaluations FE cannot be directly compared because the termination criterion was different for the PGA. But the PGA did scale like $n\sqrt{n}$ in a much smaller range of n. Thus the performance of the BGA gets better compared to the PGA, the larger the problem size n is. This example shows the advantage of investigating the scaling of heuristic search methods.

$$F_6(x) = nA + \sum_1^n (x_i^2 - A\ cos(2\pi x_i)) \qquad -5.12 \le x_i \le 5.12$$

$$F_7(x) = \sum_1^n -x_i\ sin\left(\sqrt{|x_i|}\right) \qquad -500 \le x_i \le 500$$

$$F_8(x) = \sum_1^n \frac{x_i^2}{4000} - \prod_1^n cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad -600 \le x_i \le 600$$

$$F_9(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^n cos(2\pi x_i)\right) + 20 + e \qquad -30 \le x_i \le 30$$

Table 11: Test functions.

Recent applications of evolutionary strategies to function optimization are reported in (Bäck & Hoffmeister, 1991) (Eshelman & Schaffer, 1991), (Schraudolph & Belew, 1992) (Voigt, Santibanez-Koref & Born, 1992), (Born, Voigt & Santibanez-Koref, 1992). We do not make explicit comparisons here. Instead we hope that the authors investigate the scaling of their methods. This will make comparisons very simple.

9 Conclusion

The BGA is a robust global optimization method based on a solid theory. Selection, recombination and mutation are well tuned and have a synergetic effect. The only parameter to be input by the user is the size of the population.

The BGA is inspired by artificial selection as performed by human breeders. But mutation and recombination are based on mathematical search techniques. The BGA mutation scheme is able to optimize many multimodal functions. The BGA solved in this paper some of the most popular test functions in global optimization in $O(n \cdot ln(n))$ function evaluations, the same number as for unimodal functions. This result demonstrates that these test functions are not as difficult to optimize than was formerly believed.

The standard BGA is no miracle. It is not difficult to construct challenging optimization problems for it. These problems have deep and narrow valleys of unknown directions. Progress is made only in following these valleys. Line recombination is a good search operator for such a problem. But are such problems typical applications? We did not yet encounter such a problem.

The BGA has been successfully applied to a number of real world applications. The largest application so far was the determination of discriminance functions of 560 variables for pattern recognition. The BGA solved this problem easily.

The BGA will be extended in two directions. First, the virtual breeder will continuously monitor its population and take appropriate actions. Second, more genetic operators will be implemented. This means that our virtual breeder may use "biotechnology" to improve the progress of its population. The genetic operators will be tested in parallel in different subpopulations. The operators which give good

results will grow in the total population. This framework has been implemented in our Distributed Breeder Genetic Algorithm.

References

Ackley, D. (1987). A Connectionist Machine for Genetic Hillclimbing. Kluver Academic Publisher, Boston.

Bäck, Th. & Hoffmeister, F. (1991). Extended selection mechanism in genetic algorithms. In R.K. Belew and L.B. Booker (Eds.), *Fourth Int. Conf. on Genetic Algorithms* (pp. 92–99). San Mateo, CA: Morgan Kaufmann.

Bäck, Th., Hoffmeister, F. & Schwefel. H.P. (1991). A survey of evolution strategies. In R.K. Belew and L.B. Booker (Eds.), *Fourth Int. Conf. on Genetic Algorithms* (page 2). San Mateo, CA: Morgan Kaufmann.

Bremermann, H. J., Rogson, J. and Salaff, S. (1966). Global properties of evolution processes. In Pattee, H. H. (Ed.), *Natural Automata and Useful Simulations* (pp. 3–42).

Bulmer, M.G. (1980). *"The Mathematical Theory of Quantitative Genetics"*. Clarendon Press, Oxford.

Born, J., Voigt, H.- M. & Santibanez-Koref, I. (1992). Alternative Strategies to Global Optimization In Männer, R. & Manderick, B. (Eds.), *Parallel Problem Solving from Nature* (pp. 15–26), Amsterdam: North-Holland.

Crow, J. F. (1986). *Basic Concepts in Population, Quantitative and Evolutionary Genetics.* Freeman, New York.

Crow, J. F. & Kimura, M. (1970). *An Introduction to Population Genetics Theory.* Harper and Row, New York.

DeJong, K. A. (1992). Are Genetic Algorithms Function Optimizers? In Männer, R. & Manderick, B., (Eds.), *Parallel Problem Solving from Nature* (pp. 3–13), Amsterdam: North-Holland.

Eshelman, L. J. & Schaffer, J. D. (1991). Preventing premature convergence in genetic algorithms by preventing incest. In Belew, R.K. & Booker, L.B. (Eds.), *Fourth Int. Conf. on Genetic Algorithms* (pp. 115–122), San Mateo, CA: Morgan Kaufmann.

Falconer, D. S. (1981). *Introduction to Quantitative Genetics.* Longman, London.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

Goldberg, D.E. & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G. (Ed.), *Foundations of Genetic Algorithms* (pp.

69–93), San Mateo, CA, Morgan-Kaufman.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16, 122–128.

Mühlenbein, H. (1991). Evolution in time and space - the parallel genetic algorithm. In G. Rawlins (Ed.), *Foundations of Genetic Algorithms* (pp. 316–337), San Mateo, CA: Morgan-Kaufman.

Mühlenbein, H. (1992a). How Genetic Algorithms Really Work: Mutation and Hill-climbing. In Männer, R. & Manderick, B. (Eds.), *Parallel Problem Solving from Nature* (pp. 15–26), Amsterdam: North-Holland.

Mühlenbein, H. (1992b). Parallel Genetic Algorithms in Combinatorial Optimization. In Balci, O., Sharda, R. & Zenios, S. (Eds.), *Computer Science and Operations Research* (pp. 441–456), New York: Pergamon Press.

Mühlenbein, H., Gorges-Schleuter, M. &, Krämer, O. (1988). Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7, 65–88.

Mühlenbein, H. & Schlierkamp-Voosen, D. (1992a). Evolutionary Algorithms: Theory and Applications. In Aarts, E.H.L & Lenstra, J.K. (Eds.), *Local Search in Combinatorial Optimization* (to be published) New York: Wiley. also *Technical Report 92-123, GMD.*

Mühlenbein, H. & Schlierkamp-Voosen, D. (1992b). The Distributed Breeder Algorithm: III. Migration. *Technical Report 92-122, GMD.*

Mühlenbein, H., Schomisch, M. & Born, J. (1991). The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17, 619–632.

Rechenberg, I. (1973). *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Information.* Fromman Verlag, Freiburg.

Robertson, A. (1970). A theory of limits in artificial selection with many loci. In Kojima, L. (Ed.), *Mathematical Topics in population Genetics* (pp. 246–288), New York: Springer.

Schraudolph, N. N. & Belew, R. K. (1992). Dynamic parameter encoding for genetic algorithms. *Machine Learning*, 9, 9–21.

Schwefel, H. P. (1981). *Numerical Optimization of Computer Models.* Wiley, Chichester.

Schumer, M.A. & Steiglitz, K. (1968). Adaptive step size random search. *IEEE Trans. Automatic Control*, AC-13, 270–276.

Solis, F.J. & Wets, R.J.-B (1981). Minimization by random search techniques. *Math. of Op. Research*, 6, 19–30.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In Schaffer, H. (Ed.), *Third Int. Conf. on Genetic Algorithms* (pp. 2–9), San Mateo, CA: Morgan

Kaufmann.

Törn, A. & Zilinskas, A. (1989). *Global Optimization.* Springer-Verlag, New York.

Voigt, H.-M., Santibanez-Koref, I. & Born, J. (1992). Hierachically Structured Distributed Genetic Algorithms. In Männer, R. & Manderick, B. (Eds.), *Parallel Problem Solving from Nature* (pp 155–164), Amsterdam: North-Holland.