

Analysis of Selection, Mutation and Recombination in Genetic Algorithms

Heinz Mühlenbein and Dirk Schlierkamp-Voosen

GMD Schloß Birlinghoven
D-53754 Sankt Augustin, Germany

Abstract. Genetic algorithms have been applied fairly successful to a number of optimization problems. Nevertheless, a common theory why and when they work is still missing. In this paper a theory is outlined which is based on the science of plant and animal breeding. A central part of the theory is the response to selection equation and the concept of heritability. A fundamental theorem states that the heritability is equal to the regression coefficient of parent to offspring. The theory is applied to analyze selection, mutation and recombination. The results are used in the Breeder Genetic Algorithm whose performance is shown to be superior to other genetic algorithms.

1 Introduction

Evolutionary algorithms which model natural evolution processes were already proposed for optimization in the 60's. We cite just one representative example, the outstanding work of Bremermann. He wrote in [6]. "The major purpose of the work is the study of the effects of mutation, mating, and selection on the evolution of genotypes in the case of non-linear fitness functions. In view of the mathematical difficulties involved, computer experimentation has been utilized in combination with theoretical analysis... In a new series of experiments we found evolutionary schemes that converge much better, but with no known biological counterpart."

These remarks are still valid. The designer of evolutionary algorithms should be inspired by nature, but he should not intend a one-to-one copy. His major goal should be to develop powerful optimization methods. An optimization is powerful if it is able to solve difficult optimization problems. Furthermore the algorithm should be based on a solid theory. We object popular arguments along the lines: "This is a good optimization method because it is used in nature", and vice versa: "This cannot be a good optimization procedure because you do not find it in nature".

Modelling the evolution process and applying it to optimization problems is a challenging task. We see at least two families of algorithms, one modelling

⁴⁷¹¹ In: Wolfgang Banzhaf and Frank H. Eeckman, Eds., *Evolution as a Computational Process*, Lecture Notes in Computer Science, pages 188–214, Springer, Berlin, 1995

natural and self-organized evolution, the other is based on rational selection as done by human breeders. In principle artificial selection of animals for breeding and artificial selection of virtual animals on a computer is the same problem. Therefore the designer of an evolutionary algorithm can profit from the knowledge accumulated by human breeders. But in the course of applying the algorithm to difficult fitness landscapes, the human breeder may also profit from the experience gained by applying the algorithm.

Bremermann notes [6]: “One of the results was unexpected. The evolution process may stagnate far from the optimum, even in the case of a smooth convex fitness function...It can be traced to the bias that is introduced into the sampling of directions by essentially mutating one gene at a time. One may think that mating would offset this bias; however, in many experiments mating did little to improve convergence of the process.”

Bremermann used the term mating for recombining two (or more) parent strings into an offspring. The *stagnation problem* will be solved in this paper. Bremermann’s algorithm contained most of the ingredients of a good evolutionary algorithm. But because of limited computer experiments and a missing theory, he did not find a good combination of the ingredients.

In the 70’s two different evolutionary algorithms independently emerged - the genetic algorithm of Holland [18] and the evolution strategies of Rechenberg [24] and Schwefel [27]. Holland was not so much interested in optimization, but in adaptation. He investigated the genetic algorithm with decision theory for discrete domains. Holland emphasized the importance of recombination in large populations, whereas Rechenberg and Schwefel mainly investigated normally distributed mutations in very small populations for continuous parameter optimization.

Evolutionary algorithms are random search methods which can be applied to both discrete and continuous functions. In this paper the theory of evolutionary algorithms will be based on the answers to the following questions:

- Given a population, how should the selection be done?
- Given a mutation scheme, what is the expected progress of successful mutations?
- Given a selection and recombination schedule, what is the expected progress of the population?
- How can selection, mutation and recombination be combined in synergistic manner?

This approach is opposite to the standard GA analysis initiated by Holland, which starts with the schema theorem [18]. The theorem predicts the effect of proportionate selection. Later mutation and recombination are introduced as disruptions of the population. Our view is the opposite. We regard mutation and recombination as constructive search operators. They have to be evaluated according to the probability that they create better solutions.

The search strategies of mutation and recombination are different. Mutation is based on chance. It works most efficiently in small populations. The progress

for a single mutation step is almost unpredictable. Recombination is a more global search based on restricted chance. The bias is implicitly given by the population. Recombination only shuffles the substrings contained in the population. The substrings of the optimum have to be present in the population. Otherwise a search by recombination is not able to locate the optimum.

Central themes of plant and animal breeding as well as of genetic algorithms can be phrased in statistical terms and can make substantial use of statistical techniques. In fact, problems of breeding have been the driving forces behind the development of statistics early in this century. The English school of biometry introduced a variety of now standard statistical techniques, including those of correlation and regression. We will use these techniques in order to answer the above questions. A central role plays the *response to selection* equation developed in quantitative genetics.

The outline of the paper is as follows. In section 2 some popular evolutionary algorithms are surveyed. Truncation selection and proportionate selection are investigated in section 3. In section 4 a fundamental theorem is proven which connects the response to selection equation with parent-offspring regression. Recombination/crossover and mutation are theoretically analyzed in sections 5 and 6. In section 7 mutation vs. crossover is investigated by means of a competition between these two strategies. Then numerical results are given for a test suite of discrete functions.

2 Evolutionary Algorithms

A previous survey of search strategies based on evolution has been done in [20]. Evolutionary algorithms for continuous parameter optimization are surveyed in [4].

Algorithms which are driven mainly by mutation and selection have been developed by Rechenberg [24] and Schwefel [27] for continuous parameter optimization. Their algorithms are called evolution strategies.

$(\mu + \lambda)$ Evolution Strategy

STEP1: Create an initial population of size λ

STEP2: Compute the fitness $F(x_i) \quad i = 1, \dots, \lambda$

STEP3: Select the $\mu < \lambda$ best individuals

STEP4: Create λ/μ offspring of each of the μ individuals by small variation

STEP5: If not finished, return to STEP2

An evolution strategy is a random search which uses selection and variation. The small variation is done by randomly choosing a number of a normal distribution with zero mean. This number is added to the value of the continuous variable. The algorithm adapts the amount of variation by changing the variance of the normal distribution. The most popular algorithm uses $\mu = \lambda = 1$

In biological terms, evolution strategies model natural evolution by asexual reproduction with mutation and selection. Search algorithms which model sexual

reproduction are called genetic algorithms. Sexual reproduction is characterized by recombining two parent strings into an offspring. The recombination is called *crossover*. Genetic algorithms were invented by Holland [18]. Recent surveys can be found in [14] and the proceedings of the international conferences on genetic algorithms [25] [5] [13].

Genetic Algorithm

- STEP0:** Define a genetic representation of the problem
- STEP1:** Create an initial population $P(0) = x_1^0, \dots, x_N^0$
- STEP2:** Compute the average fitness $\bar{F} = \sum_i^N F(x_i)/N$. Assign each individual the normalized fitness value $F(x_i^t)/\bar{F}$
- STEP3:** Assign each x_i a probability $p(x_i, t)$ proportional to its normalized fitness. Using this distribution, select N vectors from $P(t)$. This gives the set $S(t)$
- STEP4:** Pair all of the vectors in $S(t)$ at random forming $N/2$ pairs. Apply crossover with probability p_{cross} to each pair and other genetic operators such as mutation, forming a new population $P(t+1)$
- STEP5:** Set $t = t + 1$, return to STEP2

In the simplest case the genetic representation is just a bitstring of length n , the *chromosome*. The positions of the strings are called *loci* of the chromosome. The variable at a locus is called *gene*, its value *allele*. The set of chromosomes is called the *genotype* which defines a *phenotype* (the individual) with a certain fitness.

The genetic operator *mutation* changes with a given probability p_m each bit of the selected string. The *crossover* operator works with two strings. If two strings $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ are given, then the *uniform crossover* operator [28] combines the two strings as follows

$$z = (z_1, \dots, z_n) \quad z_i = x_i \text{ or } z_i = y_i$$

Normally x_i or y_i are chosen with equal probability.

In genetic algorithms many different crossover operators are used. Most popular are one-point and two-point crossover. One or two loci of the string are randomly chosen. Between these loci the parent strings are exchanged. This exchange models crossover of chromosomes found in nature. The disruptive uniform crossover is not used in nature. It can be seen as n -point crossover.

The crossover operator links two probabilistically chosen searches. The information contained in two strings is mixed to generate a new string. Instead of crossing-over I prefer to use the general term *recombination* for any method of combining two or more strings.

A genetic algorithm is a parallel random search with centralized control. The centralized part is the selection schedule. The selection needs the average fitness of the population. The result is a highly synchronized algorithm, which is difficult to implement efficiently on parallel computers. In the *parallel genetic algorithm PGA* [20],[21], a distributed selection scheme is used. This is achieved

as follows. Each individual does the selection by itself. It looks for a partner in its neighborhood only. The set of neighborhoods defines a spatial population structure.

The second major change can also easily be understood. Each individual is active and not acted on. It may improve its fitness during its lifetime by performing a local search.

The parallel genetic algorithm PGA can be described as follows: :

Parallel Genetic Algorithm

- STEP0:** Define a genetic representation of the problem
- STEP1:** Create an initial population and its population structure
- STEP2:** Each individual does local hill-climbing
- STEP3:** Each individual selects a partner for mating in its neighborhood
- STEP4:** An offspring is created with genetic operators working on the genotypes of its parents
- STEP5:** The offspring does local hill-climbing. It replaces the parent, if it is better than some criterion (acceptance)
- STEP6:** If not finished, return to STEP3.

It has to be noticed that each individual may use a different local hill-climbing method. This feature will be important for problems, where the efficiency of a particular hill-climbing method depends on the problem instance.

In the PGA the information exchange within the whole population is a diffusion process because the neighborhoods of the individuals overlap. All decisions are made by the individuals themselves. Therefore the PGA is a totally distributed algorithm without any central control. The PGA models the *natural evolution process* which self-organizes itself.

The next algorithm, the *breeder genetic algorithm BGA* [22] is inspired by the science of breeding animals. In this algorithm, each one of a set of virtual breeders has the task to improve its own subpopulation. Occasionally the breeder imports individuals from neighboring subpopulations. The DBGA models *rational controlled evolution*. We will describe the breeder genetic algorithm only.

Breeder Genetic Algorithm

- STEP0:** Define a genetic representation of the problem
- STEP1:** Create an initial population $P(0)$
- STEP2:** Each individual may perform local hill-climbing
- STEP3:** The breeder selects $T\%$ of the population for mating. This gives set $S(t)$
- STEP4:** Pair all the vectors in $S(t)$ at random forming N pairs. Apply the genetic operators crossover and mutation, forming a new population $P(t + 1)$.
- STEP5:** Set $t = t + 1$, return to STEP2 if it is better than some criterion (acceptance)
- STEP6:** If not finished, return to STEP3.

The major difference between the genetic algorithm and the breeder genetic algorithm is the method of selection. The breeders have developed many different selection strategies. We only want to mention *truncation selection* which breeders usually apply for large populations. In truncation selection the $T\%$ best individuals of a population are selected as parents.

The different evolutionary algorithms described above put different emphasis on the three most important evolutionary forces, namely selection, mutation and recombination. We will in the next sections analyze these evolutionary forces by methods developed in *quantitative genetics*. One of the most important aspect of algorithms inspired by processes found in nature is the fact that they can be investigated by the methods proven usefully in the natural sciences.

3 Natural vs. Artificial Selection

The theoretical analysis of evolution centered in the last 60 years on understanding evolution in a natural environment. It tried to model *natural selection*. The term natural selection was informally introduced by Darwin in his famous book "On the origins of species by means of natural selection". He wrote: "The preservation of favourable variations and the rejection of injurious variations, I call Natural Selection." Modelling natural selection mathematically is difficult. Normally biologist introduce another term, the fitness of an individual which is defined as the number of offspring of that individual. This fitness definition cannot be used for prediction. It can only be measured after the individual is not able to reproduce any more. *Artificial selection* as used by breeders is seldom investigated in textbooks on evolution. It is described in more practical books aimed for the breeders. We believe that this is a mistake. Artificial selection is a controlled experiment, like an experiment in physics. It can be used to isolate and understand specific aspects of evolution. Individuals are selected by the breeder according to some trait. In artificial selection predicting the outcome of a breeding programme plays a major role.

Darwin recognized the importance of artificial selection. He devoted the whole first chapter of his book to artificial selection by breeders. In fact, artificial selection independently done by a number of breeders served as a model for natural selection. Darwin wrote: "I have called this principle by the term Natural Selection in order to mark its relation to man's power of selection."

In this section we will first analyze artificial selection by methods found in quantitative genetics [11], [8] and [7]. A mathematically oriented book on quantitative genetics and natural selection is [9]. We will show at the end of this section that natural selection can be investigated by the same methods. A detailed investigation can be found in [23].

3.1 Artificial Selection

The change produced by selection that mainly interests the breeder is the *response to selection*, which is symbolized by R . R is defined as the difference be-

tween the population mean fitness $M(t+1)$ of generation $t+1$ and the population mean of generation t . $R(t)$ estimates the expected progress of the population.

$$R(t) = M(t+1) - M(t) \quad (1)$$

Breeders measure the selection with the *selection differential*, which is symbolized by S . It is defined as the difference between the average fitness of the selected parents and the average fitness of the population.

$$S(t) = M_s(t) - M(t) \quad (2)$$

These two definitions are very important. They quantify the most important variables. The breeder tries to predict $R(t)$ from $S(t)$. Breeders often use *truncation selection* or *mass selection*. In truncation selection with threshold $Trunc$, the $Trunc$ % best individuals will be selected as parents. $Trunc$ is normally chosen in the range 50% to 10%.

The prediction of the response to selection starts with

$$R(t) = b_t \cdot S(t) \quad (3)$$

b_t is called the *realized heritability*. The breeder either measures b_t in previous generations or estimates b_t by different methods [23]. It is normally assumed that b_t is constant for a certain number of generations. This leads to

$$R(t) = b \cdot S(t) \quad (4)$$

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.

$$R_n = \sum_{t=1}^n R(t) \quad (5)$$

In order to compute R_n a second equation is needed. In quantitative genetics, several approximate equations for $S(t)$ are proposed [7], [11]. Unfortunately these equations are only valid for *diploid* organisms. Diploid organisms have two sets of chromosomes. Most genetic algorithms use one set of chromosomes, i.e. deal with *haploid* organisms. Therefore, we can only apply the research methods of quantitative genetics, not the results.

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

$$S = I \cdot \sigma_p \quad (6)$$

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 [7]. In table 1 the relation between the truncation threshold $Trunc$ and the selection intensity I is shown. A decrease from 50 % to 1 % leads to an increase of the

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

Table 1. Selection intensity.

selection intensity from 0.8 to 2.66.

If we insert (6) into (4) we obtain the well-known *response to selection equation* [11].

$$R(t) = b \cdot I \cdot \sigma_p(t) \quad (7)$$

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

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. We assume *uniform crossover* for recombination. For the simple *ONEMAX* function 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. But this 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 crossover 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

$$\sigma_p(t) = \sqrt{n \cdot p(t) \cdot (1 - p(t))} \quad (8)$$

Theorem 1. *If the population is large enough that it converges to the optimum and if the selection intensity I is greater than 0, then the response to selection is given for the ONEMAX function by*

$$R(t) = \frac{I}{\sqrt{n}} \cdot \sqrt{p(t)(1 - p(t))} \quad (9)$$

The number of generations needed until equilibrium is approximate

$$GEN_e = \left(\frac{\pi}{2} - \arcsin(2p_0 - 1) \right) \cdot \frac{\sqrt{n}}{I} \quad (10)$$

$p_0 = p(0)$ denotes the probability of the advantageous bit in the initial population.

Proof. Noting that $R(t) = n(p(t+1) - p(t))$ we obtain the difference equation

$$p(t+1) - p(t) = \frac{I}{\sqrt{n}} \cdot \sqrt{p(t) \cdot (1-p(t))} \quad (11)$$

The difference equation can be approximated by a differential equation

$$\frac{dp(t)}{dt} = \frac{I}{\sqrt{n}} \cdot \sqrt{p(t) \cdot (1-p(t))} \quad (12)$$

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

$$p(t) = 0.5 \left(1 + \sin \left(\frac{I}{\sqrt{n}} t + \arcsin(2p_0 - 1) \right) \right) \quad (13)$$

The convergence of the total population is characterized by $p(GEN_e) = 1$. GEN_e can be easily computed from the above equation. One obtains

$$GEN_e = \left(\frac{\pi}{2} - \arcsin(2p_0 - 1) \right) \cdot \frac{\sqrt{n}}{I} \quad (14)$$

■

The number of generations needed until convergence is proportional to \sqrt{n} and inversely proportional to the selection intensity. Note that the equations are only valid if the size of the population is large enough so that the population converges to the optimum. The most efficient breeder genetic algorithm runs with the *minimal popsize* N^* , so that the population still converges to the optimum. N^* depends on the size of the problem n , the selection intensity I and the probability of the advantageous bit p_0 . This problem will be discussed in section 5.

Remark: The above theorem assumes that the variance of the fitness is binomial distributed. Simulations show that the phenotypic variance is slightly less than given by the binomial distribution. The empirical data is better fitted if the binomial variance is reduced by a factor $\pi/4.3$. Using this variance one obtains the equations

$$\tilde{R}(t) = \frac{\pi}{4.3} \cdot \frac{I}{\sqrt{n}} \cdot \sqrt{p(t)(1-p(t))} \quad (15)$$

$$G\tilde{E}N_e = \frac{4.3}{\pi} \left(\frac{\pi}{2} - \arcsin(2p_0 - 1) \right) \cdot \frac{\sqrt{n}}{I} \quad (16)$$

Equation 15 is a good prediction for the mean fitness of the population. This is demonstrated in figure 1. The mean fitness versus the number of generations is shown for three popsizes $N = 1024, 256, 64$. The selection intensity is $I = 0.8$, the size of the problem $n = 64$. The initial population was generated with $p_0 = 1/64$. The fit of equation 15 and the simulation run with $N = 1024$ is very good. For $N = 256$ and $N = 64$ the population does not converge to the optimum. These popsizes are less than the critical popsize $N^*(I, n, p_0)$.

A more detailed evaluation of equation 15 can be found in [23].

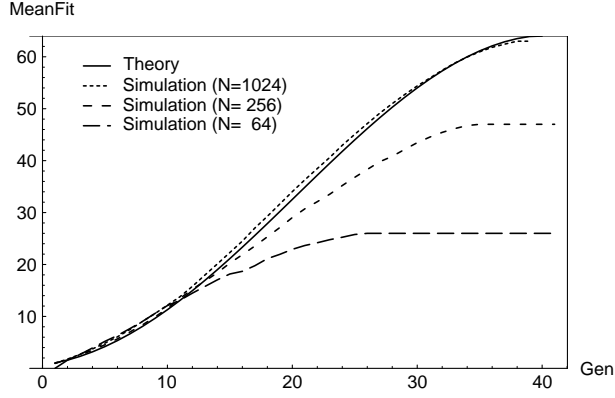


Fig. 1. Mean fitness for theory and simulations for various N

3.2 Natural Selection

Natural selection is modelled by proportionate selection in quantitative genetics. Proportionate selection is defined as follows. Let $0 \leq g_i(t) \leq 1$ be the proportion of genotype i in a population of size N at generation t , F_i its fitness. Then the phenotype distribution of the selected parents is given by

$$g_{i,S}(t) = \frac{g_i(t)F_i}{M(t)} \quad (17)$$

where $M(t)$ is the average fitness of the population

$$M(t) = \sum_{i=1}^N g_i(t)F_i \quad (18)$$

Note that proportionate selection is also used by the simple genetic algorithm [14].

Theorem 2. *In proportionate selection the selection differential is given by*

$$S(t) = \frac{\sigma_p^2(t)}{M(t)} \quad (19)$$

For the ONEMAX function of size n the response to selection is given by

$$R(t) = 1 - p(t) \quad (20)$$

If the population is large enough, the number of generations until $p(t) = 1 - \epsilon$ is given for large n by

$$GEN_{1-\epsilon} \approx n \cdot \ln \frac{1-p_0}{\epsilon} \quad (21)$$

p_0 is the probability of the advantageous allele in the initial population.

Proof.

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

For *ONEMAX*(n) we have $R(t+1) = S(t)$. Furthermore we approximate

$$\sigma_p^2(t) \approx np(t)(1-p(t)) \quad (22)$$

Because $M(t) = np(t)$, equation 20 is obtained. From $R(t) = n(p(t+1) - p(t))$ we get the difference equation

$$p(t+1) = \frac{1}{n} + (1 - \frac{1}{n})p(t) \quad (23)$$

This equation has the solution

$$p(t) = \frac{1}{n} (1 + (1 - \frac{1}{n}) + \dots + (1 - \frac{1}{n})^{t-1}) + (1 - \frac{1}{n})^t p_0$$

This equation can be simplified to

$$p(t) = 1 - (1 - \frac{1}{n})^t (1 - p_0)$$

By setting $p(\text{GEN}_{1-\epsilon}) = 1 - \epsilon$ equation 21 is easily obtained. ■

Remark: If we assume $R(t) = S(t)$ we obtain from equation 19 a version of Fisher's *fundamental theorem of natural selection* [12] [9].

By comparing truncation selection and proportionate selection one observes that proportionate selection gets weaker when the population approaches the optimum. An infinite population will need an infinite number of generations for convergence. In contrast, with truncation selection the population will converge in at most $O(\sqrt{n})$ generations independent of the size of the population. Therefore truncation selection as used by breeders is much more effective than proportionate selection for optimization.

The major results of these investigations can be summarized as follows. *A genetic algorithm using recombination/crossover only is most efficient if run with the minimal population size N^* so that the population converges to the optimum. Proportionate selection as used by the simple genetic algorithm is inefficient.*

4 Statistics and Genetics

Central themes of plant and animal breeding as well as of genetic algorithms can be phrased in statistical terms and can make substantial use of statistical techniques. In fact, problems of breeding have been the driving forces behind the development of statistics early in this century. The English school of biometry introduced a variety of now standard statistical techniques, including those of correlation and regression. In this section we will only prove the fundamental theorem, which connects the rather artificial factor $b(t)$ with the well known regression coefficient of parent-offspring.

Theorem 3. *Let $X(t) = (x_1(t), \dots, x_N(t))$ be the population at generation t , where x_i denotes the phenotypic value of individual i . Assume that an offspring generation $X'(t+1)$ is created by random mating, without selection. If the regression equation*

$$x'_{ij}(t+1) = a(t) + b_{X'X}(t) \cdot \frac{x_i(t) + x_j(t)}{2} + \epsilon_{ij} \quad (24)$$

with

$$E(\epsilon_{ij}) = 0$$

is valid, where x'_{ij} is the offspring of x_i and x_j , then

$$b_{X'X}(t) \approx b(t) \quad (25)$$

Proof. From the regression equation we obtain for the averages

$$E(x'(t+1)) = a(t) + b_{X'X}(t)M(t)$$

Because the offspring generation is created by random mating without selection, the expected average fitness remains constant

$$E(x'(t+1)) = M(t)$$

Let us now select a subset $X_S(t) \subset X(t)$ as parents. The parents are randomly mated, producing the offspring generation $X(t+1)$. If the subset $X_S(t)$ is large enough, we may use the regression equation and get for the averages

$$E(x(t+1)) = a(t) + b_{X'X}(t) * (M_s(t) - M(t))$$

Subtracting the above equations we obtain

$$M(t+1) - M(t) = b_{X'X}(t)S(t)$$

■

For the proof we have used some additional statistical assumptions. It is outside the scope of this paper to discuss these assumptions in detail.

The problem of computing a good regression coefficient is solved by the theorem of Gauss-Markov. The proof can be found in any textbook on statistics.

Theorem 4. *A good estimate for the regression coefficient is given by*

$$b_{X'X}(t) = 2 * \frac{cov(x'(t), x(t))}{var(x(t))} \quad (26)$$

These two theorems allow the estimation of the factor $b(t)$ without doing a selection experiment. In quantitative genetics $b(t)$ is called the *heritability* of the trait to be optimized. We have shown in [23] how to apply these theorems to the breeder genetic algorithm.

5 Analysis of recombination and selection

In this section we will make a detailed analysis of selection and crossover by simulations. First we will explain the performance of the crossover operator in finite populations by a diagram. We will use *ONEMAX* as fitness function. In figure 2 the number of generations GEN_e until equilibrium and the size of the population are displayed. At equilibrium the whole population consists of one genotype only. The initial population was randomly generated with probability $p_0 = 0.2$ of the advantageous allele. The data are averages over 100 runs.

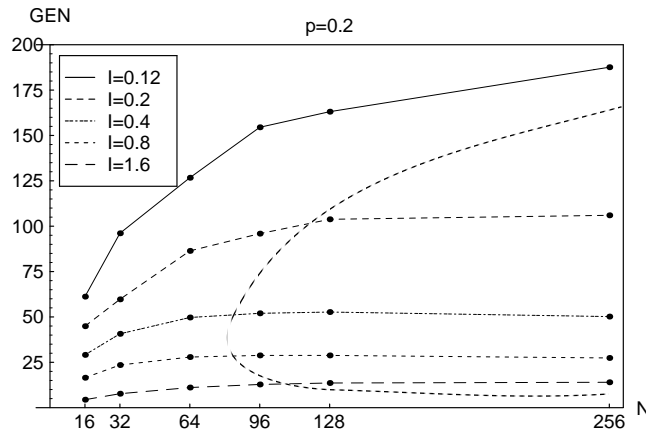


Fig. 2. GEN_e vs population size N for $p_0 = 0.2$ and $p_0 = 0.5$

The figure can be divided into three areas. The first area we name *saturation region*. The population size is large enough so that the population converges to

the optimum value. In this area GEN_e is constant. This is an important result, because it is commonly believed in population genetics that GEN_e increases with the population size [19]. This is only the case in the second region. Here the population size is too small. The population does not converge to the optimum. GEN_e increases with the population size because the quality of the final solution gets better.

The two regions are separated by the *critical population size* N^* . It is the minimal population size so that the population converges to the optimum. N^* depends on the selection intensity I , the size of the problem and the initial population. The relation between N^* and I is especially difficult. N^* increases for small selection intensities I and for large ones. The increase for large I can be easily understood. If only one individual is selected as parent, then the population converges in one generation. In this case the genotype of the optimum has to be contained in the initial population. So the population size has to be very large.

The increase of N^* with small selection intensity is more difficult to understand. It is related to the *genetic drift*. It has been known for quite a time that the population converges also without any kind of selection just because of random sampling in a finite population. In [1] it has been shown that GEN_e increases proportional to the size of the population N and to the logarithm of the size of the problem n . Thus GEN_e is surprisingly small.

This important result demonstrates that *chance alone is sufficient to drive a finite population to an equilibrium*. The formula has been proven for one gene in [9]. It led to the development of the *neutral theory* of evolution [19]. This theory states that many aspects of natural evolution can be explained by neutral mutations which got fixed because of the finite population size. Selection seems to be not as important as previously thought for explaining natural evolution.

We are now able to understand why N^* has to increase for small selection intensities. The population will converge in a number of generations proportional to the size of the population. Therefore the size of the population has to be large enough that the best genotype is randomly generated during this time.

From GEN_e the number of trials till convergence can be easily computed by

$$FE_e = N \cdot GEN_e$$

In order to minimize FE_e , the BGA should be run with the minimal popsize $N^*(I, n, p_0)$. The problem of predicting N^* is very difficult because the transition from region 2 to the saturation region is very slow. In this paper we will only make a qualitative comparison of mutation and crossover. Therefore a closed expression for N^* is not needed. In [23] some formulas for N^* are derived.

The major results of this section can be summarized as follows: *A genetic algorithms with recombination/crossover is only effective in large populations. It runs most efficiently with the critical population size $N^*(I, n, p_0)$. The response to selection can be accurately predicted for the saturation region.*

6 Analysis of Mutation

The mutation operator in small populations is well understood. The analysis of mutation in large populations is more difficult. In principle it is just a problem of statistics - doing N trials in parallel instead of a sequence. But the selection converts the problem to a nonstandard statistical problem. We will solve this problem by an extension of the response to selection equation.

In [21] we have computed the probability of a successful mutation for a single individual. From this analysis the optimal mutation rate has been obtained. The optimal mutation rate maximizes the probability of a success. We just state the most important results.

Theorem 5. *For the ONEMAX function of size n the optimal mutation rate m is proportional to the size of the problem.*

$$m = \frac{1}{n}$$

This important result has been independently discovered several times. The implications of this result to biology and to evolutionary algorithms have been first investigated by Bremermann [6].

The performance of crossover was measured by GEN_e , the number of generations until equilibrium. This measure cannot be used for mutation because the population will never converge to a unique genotype. Therefore we will use as performance measure for mutation GEN_{opt} . It is defined as the average number of generations till the optimum has been found for the first time. For a population with two individuals (one parent and one offspring) GEN_{opt} has been computed by a Markov chain analysis [21]. In this case GEN_{opt} is equal to FE_{opt} , the number of trials to reach the optimum.

Theorem 6. *Let p_0 be the probability of the advantageous allele in the initial string. Then the (1+1) evolutionary algorithm needs on the average the following number of trials FE_{opt}*

$$FE_{opt} = e \cdot n \sum_{j=1}^{(1-p_0)n} \frac{1}{j} \quad (27)$$

to reach the optimum. The mutation rate is set to $m = 1/n$.

Proof. We only sketch the proof. Let the given string have one incorrect bit left. Then the probability of switching this bit is given by

$$s_1 = m * (1 - m)^{n-1} \approx e^{-1} \cdot m \quad (28)$$

The number of trials to obtain the optimum is given by $e * 1/m$. Similarly if two bits are incorrect, then the number of trials needed to get one bit correct is given by $e/2 * 1/m$. The total number is obtained by summation. ■

For $0 \leq p_0 < 0.9$ the above equation can be approximated by

$$FE_{opt} = e \cdot n \cdot \ln((1 - p_0)n) \quad (29)$$

We have confirmed the formula by intensive simulations [21]. Recently Bäck [2] has shown that FE_{opt} can be only marginally reduced if a theoretically optimal *variable* mutation rate is used. This mutation rate depends on the number of bits which are still wrong. This result has been predicted in [21]. Mutation spends most of the time in adjusting the very last bits. But in this region the optimal mutation rate is $m = 1/n$.

Next we will extend the analysis to large populations. First we will use simulation results. In figure 3 the relation between GEN_{opt} , FE_{opt} , and the popsize N is displayed for two selection methods. The selection thresholds are $T = 50\%$ and the smallest one possible, $T = 1/N$. In the latter case only the best individual is selected as parent. In large populations the strong selection outperforms the fixed selection scheme by far. These results can easily be explained. The mutation operator will change one bit on the average. The probability of a success gets less the nearer the population comes to the optimum. Therefore the best strategy is to take just the best individual as parent of the next generation.

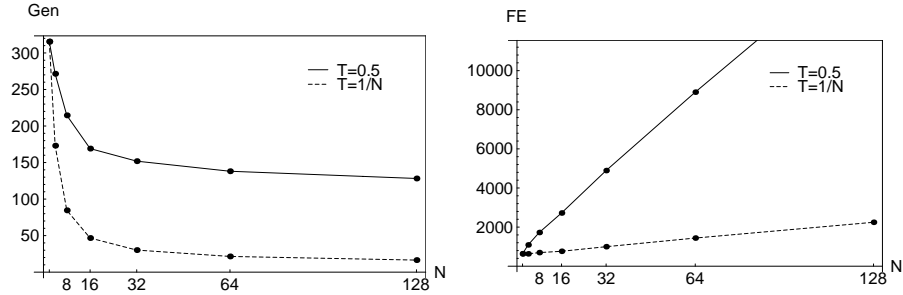


Fig. 3. GEN_{opt} and function evaluations (FE) for various N and different T

From GEN_{opt} the expected number of trials needed to find the optimum can be computed

$$FE_{opt} = N \cdot GEN_{opt}$$

For both selection methods, FE_{opt} increases linearly with N for large N . The increase is much smaller for the strong selection. The smallest number of function evaluations are obtained for $N = 1, 2, 4$.

We now turn to the theoretical analysis. It depends on an extension of the response to selection equation.

Theorem 7. *Let u_t be the probability of a mutation success, imp the average improvement of a successful mutation. Let v_t be the probability that the offspring*

is worse than the parent, red the average reduction of the fitness. Then the response to selection for small mutations in large populations is given by

$$R(t) = S(t) + u_t \cdot imp - v_t \cdot red \quad (30)$$

$S(t)$ is the average fitness of the selected parents.

Proof. Let $M_s(t)$ be the average of the selected parents. Then

$$M(t+1) = u_t(M_s(t) + imp) + v_t(M_s(t) - red) + (1 - u_t - v_t)M_s(t)$$

Subtracting $M(t)$ from both sides of the equation we obtain the theorem. ■

The response to selection equation for mutation contains no *heritability*. Instead there is an offset, defined by the difference of the probabilities of getting better or worse. The importance of u_t and v_t has been independently discovered by Schaffer et al. [26]. They did not use the difference of the probabilities, but the quotient which they called the *safety factor*.

$$F = \frac{u_t}{v_t}$$

In order to apply the theorem we have to estimate $S(t)$, u_t and v_t . The last two variables can be estimated by using the results of [21]. The estimation needs the average number i of wrong bits of the parent strings as input. But i can be easily transformed into a variable depending on the state of the population at generation t . This variable is the marginal probability $p(t)$ that there is the advantageous allele at a locus. $p(t)$ was already used in the previous theorems. i and $p(t)$ are connected by

$$i \approx n \cdot (1 - p(t)) = n - M(t) \quad (31)$$

We have been not able to estimate $S(t)$ analytically. For the next result we have used simulations. Therefore we call it an empirical law.

Empirical Law 1 *For the ONEMAX function, a truncation threshold of $T = 50\%$, a mutation rate of $m = 1/n$, and $n \gg 1$ the response to selection of a large population changing by mutation is approximate*

$$R(t) = 1 + (1 - p(t))e^{-p(t)} - p(t)e^{-(1-p(t))} \quad (32)$$

Proof. Let the parents have i bits wrong, let s_i be the probability of a success by mutation, f_i be the probability of a defect mutation. s_i is approximately given by the product of changing at least one of the wrong bits and not changing the correct bit [21]. Therefore

$$s_i = (1 - m)^{n-i}(1 - (1 - m)^i)$$

Similarly

$$f_i = (1 - m)^i(1 - (1 - m)^{n-i})$$

From equation 31 and $1 - (1 - m)^i \approx i * m$ we obtain

$$s_t = (1 - p(t)) \left(1 - \frac{1}{n}\right)^{np(t)}$$

$$f_t = p(t) \left(1 - \frac{1}{n}\right)^{n(1-p(t))}$$

Because $\left(1 - \frac{1}{n}\right)^n \approx e^{-1}$ we get

$$s_t = (1 - p(t)) e^{-p(t)}$$

$$f_t = p(t) e^{-(1-p(t))}$$

We are left with the problem to estimate *imp* and *red*. In a first approximation we set both to 1 because a mutation rate of $m = 1/n$ changes one bit on the average. We have not been able to estimate $S(t)$ analytically. Simulations show that for $T = 50\%$ $S(t)$ decreases from about 1.15 at the beginning to about 0.9 at GEN_{opt} . Therefore $S(t) = 1$ is a reasonable approximation. This completes the proof. ■

Equation 32 defines a difference equation for $p(t + 1)$. We did not succeed to solve it analytically. We have found that the following linear approximation gives almost the same results

Empirical Law 2 *Under the assumptions of empirical law 1 the response to selection can be approximated by*

$$R(t) = 2 - 2p(t) \tag{33}$$

The number of generations until $p(t) = 1 - \epsilon$ is reached is given by

$$GEN_{1-\epsilon} \approx \frac{n}{2} \cdot \ln \frac{1-p_0}{\epsilon} \tag{34}$$

Proof. The proof is identical to the proof of theorem 2.

In figure 4 the development of the mean fitness is shown. The simulations have been done with two popsizes ($N = 1024, 64$) and two mutation rates ($m = 1/n, 4/n$). The agreement between the theory and the simulation is very good. The evolution of the mean fitness of the large population and the small population is almost equal. This demonstrates that for mutation a large population is inefficient.

A large mutation rate has an interesting effect. The mean fitness increases faster at the beginning, but it never finds the optimum. This observation again suggests to use a variable mutation rate. But we have already mentioned that the increase in performance by using a variable mutation rate will be rather small. Mutation spends most of its time in getting the last bits correct. But in this region a mutation rate of $m = 1/n$ is optimal.

The major results of this section can be summarized as follows: *Mutation in large populations is not effective. It is more efficient with very strong selection. The response to selection becomes very small when the population is approaching the optimum. The efficiency of the mutation operator critically depends on the mutation rate.*

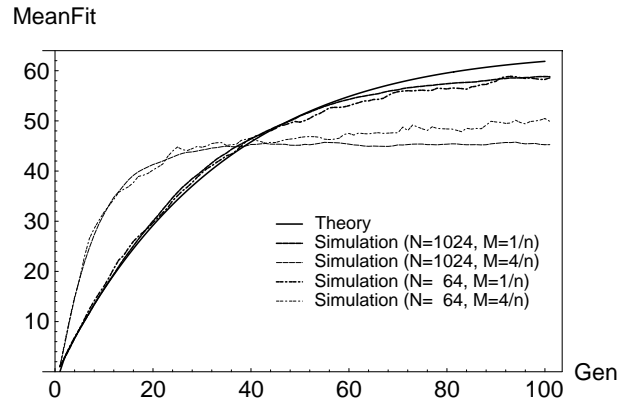


Fig. 4. Mean fitness for theory and simulations for various N and mutation probabilities

7 Competition between Mutation and Crossover

The previous sections have qualitatively shown that the crossover operator and the mutation operator are performing good in different regions of the parameter space of the BGA. In figure 5 crossover and mutation are compared quantitatively for a popsize of $N = 1024$. The initial population was generated with $p_0 = 1/64$. The mean fitness of the population with mutation is larger than that of the population with crossover until generation 18. Afterwards the population with crossover performs better. This was predicted by the analysis.

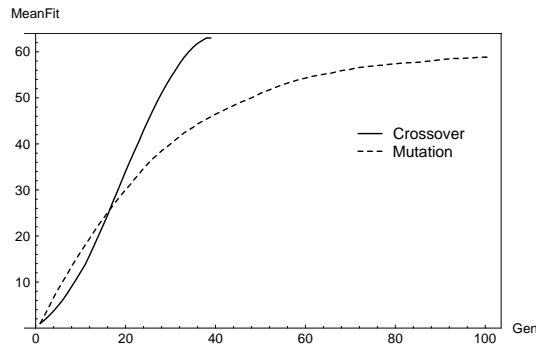


Fig. 5. Comparison of mutation and crossover

The question now arises how to best combine mutation and crossover. This can be done by two different methods at least. First one can try to use both operators in a single genetic algorithm with their optimal parameter settings. This means that a good mutation rate and a good population size has to be predicted. This method is used for the standard breeder genetic algorithm **BGA**. Results for popular test functions will be given later.

Another method is to apply a competition between subpopulations using different strategies. Such a competition is in the spirit of population dynamics. It is the foundation of the Distributed Breeder Genetic Algorithm.

Competition of strategies 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. [3] have implemented the adaptation of strategy parameters on the individual level. The strategy parameters of the best individuals are recombined, giving the new stepsize for the mutation. Herdy [17] uses an competition on the population level. In this case whole populations are evaluated at certain intervals. The strategies of the succesful populations proliferate, strategies in populations with bad performance die out. Our adaptation lies between these two extreme cases. The competition is done between subpopulations.

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 subgroups have a lower limit. Therefore no strategy is lost. The rationale behind this algorithm will be published separately.

In the experiments the mean fitness of the species was used as quality criterion. The isolation interval was four generations, the migration interval eight generations. The gain was four individuals. In the case of two groups the population size of the better group increases by four, the population size of the worse group decreases by four. If there are more than two groups competing, then a proportional rating is used.

Figure 6 shows a competition race between two groups, one using mutation only, the other crossing-over. The initial population was randomly generated with $p_0 = 1/64$. The initial population is far away from the optimum. Therefore first the population using mutation only grows, then the crossover population takes over. The first figure shows the mean fitness of the two groups. The migration strategy ensures that the mean fitness of both populations are almost equal.

In figure 7 competition is done between three groups using different mutation rates. At the beginning the group with the highest mutation rate grows, then both the middle and the lowest mutation rate grow. At the end the lowest mutation rate takes over. These experiments confirm the results of the previous sections.

In the next section we will compare the efficiency of a BGA using mutation, crossover and an optimal combination of both.

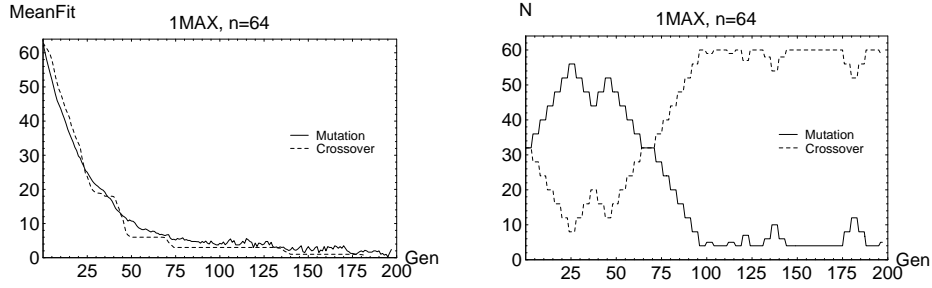


Fig. 6. Competition between mutation and crossover

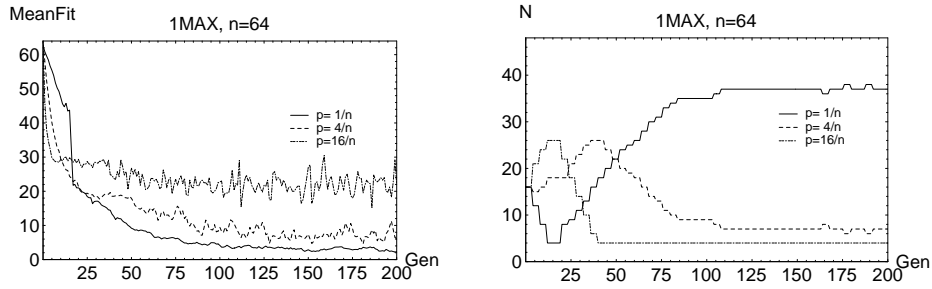


Fig. 7. Competition between different mutation rates

8 The Test Functions

The outcome of a comparison of mutation and crossover depends on the fitness landscape. Therefore a carefully chosen set of test functions is necessary. We will use test functions which we have theoretically analyzed in [21]. They are similar to the test functions used by Schaffer [26]. The test suite consists of

ONEMAX(n)
 MULTIMAX(n)
 PLATEAU(k, l)
 SYMBASIN(k, l)
 DECEPTION(k, l)

The fitness of ONEMAX is given by the number of 1's in the string. *MULTIMAX*(n) is similar to *ONEMAX*, but its global optima have exactly $n/2$ 1's contained in the string. It is defined as follows

$$MULTIMAX(n, X) = \begin{cases} \sum_{i=1}^n x_i & \sum_{i=1}^n x_i \leq n/2 \\ n - \sum_{i=1}^n x_i & \sum_{i=1}^n x_i \geq n/2 \end{cases}$$

We have included the $MULTIMAX(n)$ function in the test suite to show the dependence of the performance of the crossover operator on the fitness function. $MULTIMAX(n)$ poses no difficulty for mutation. Mutation will find one of the many global optima in $O(n)$ time. But crossover has difficulties when two different optimal strings are recombined. This will lead with high probability to a worse offspring. An example is shown below for $n = 4$

$$1100 \otimes 0011$$

With probability $P = 10/16$ will crossover create an offspring worse than the midparent. The average fitness of an offspring is $3/2$. Therefore the population will need many generations in order to converge. More precisely: The number of generations between the time when an optimum is first found and the convergence of the whole population is very high. $MULTIMAX$ is equal to $ONEMAX$ away from the global optima. In this region the heritability is one. When the population approaches the optima, the heritability drops sharply to zero. The response to selection is almost 0.

For the PLATEAU function k bits have to be flipped in order that the fitness increases by k . The DECEPTION function has been defined by Goldberg [16]. The fitness of DECEPTION(k,l) is given by the sum of l deceptive functions of size k . A deceptive function and a smoothed version of order $k = 3$ is defined in the following table

bit	DECEP	SYMBA	bit	DECEP	SYMBA
111	30	30	100	14	14
101	0	26	010	22	22
110	0	22	001	26	26
011	0	14	000	28	28

A DECEPTION function has 2^l local maxima. Neighboring maxima are k bits apart. Their fitness value differs by two. The basin of attraction of the global optimum is of size k^l , the basin of attraction of the smallest optimum is of size $(2^k - 1)^l$. The DECEPTION function is called deceptive because the search is misled to the wrong maximum $(0, 0, \dots, 0)$. The global optimum is particularly isolated.

The SYMBASIN(k,l) function is like a deceptive function, but the basins of attraction of the two peaks are equal. In the simulations we used the values given in the above table for SYMBA.

9 Numerical Results

All simulations have been done with the breeder genetic algorithm BGA. In order to keep the number of simulations small, several parameters were fixed. The mutation rate was set to $m = 1/n$ where n denotes the size of the problem. The parents were selected with a truncation threshold of $T = 35\%$. Sometimes $T = 50\%$ was used.

In the following tables the average number of generations is reported which are needed in order that the best individual is above a predefined fitness value. With these values it is possible to imagine a type of race between the populations using the different operators. Table 2 shows the results for ONEMAX of size 64. FE denotes the number of function evaluations necessary to reach the optimum. SD is the standard deviation of GEN_e if crossover is applied only. In all other cases it is GEN_{opt} , the number of generations until the optimum was found. The initial population was randomly generated with a probability $p_0 = 0.5$ that there is a 1 at a locus. The numerical values are averages over 100 runs.

OP	N	48	56	61	62	63	64	SD	FE
M	2	41	94	156	183	226	309	82	618
M	64	18	40	65	80	102	143	56	9161
C*	64	7	11	15	15	17	19	1.1	1210
C	128	5	9	12	12	13	15	10.8	1898
M&C	4	23	51	81	96	115	152	47	608
M&C	64	7	13	17	19	20	22	2.1	2102

Table 2. ONEMAX(64); C* found optimum in 84 runs only

The simulations confirm the theory. Mutation in small populations is a very effective search. But the variance SD of GEN_{opt} is very high. Furthermore, the success of mutation decreases when the population approaches the optimum. A large population reduces the efficiency of a population using mutation. Crossover is more predictable. The progress of the population is constant. But crossover critically depends on the size of the population. The most efficient search is done by the BGA using both mutation and crossover with a population size of $N = 4$.

In table 3 the initial population was generated farther away from the optimum ($p_0 = 1/8$). In this experiment, mutation in small populations is much more efficient than crossover. But the combined search is also performing good.

OP	N	24	32	62	63	64	SD	FE
M	2	14	24	192	237	307	85	615
M	64	8	16	96	117	161	72	10388
C*	256	6	9	24	25	27	0.9	6790
C	320	6	9	24	25	26	0.9	8369
M&C	4	11	19	114	136	180	52	725
M&C	64	5	8	29	31	34	3	2207

Table 3. ONEMAX(64); $P_0 = 1/8$; C* found optimum in 84 runs only

In table 4 results are presented for the PLATEAU function. The efficiency of the small population with mutation is slightly worse than for ONEMAX. But the efficiency of the large population is much better than for ONEMAX. This can be easily explained. The large population is doing a random walk on the plateau. The best efficiency has the BGA with mutation and crossover and a popsize of $N = 4$.

OP	N	288	291	294	297	300	SD	FE
M	4	27	42	64	95	184	107	737
M	64	5	8	13	19	31	9	2064
C*	64	3	4	6	7	9	1	569
C	128	3	4	5	6	8	1	1004
M&C	4	22	32.5	49	73	134	63	539
M&C	64	10	10	10	10	12	2	793

Table 4. PLATEAU(3,10); C* found optimum in 78 runs only

In table 5 results are shown for the *DECEPTION*(3, 10) function.

OP	N	283	291	294	297	300	SD	FE
M	4	419	3520	4721	6632	9797	4160	39192
M	16	117	550	677	827	1241	595	19871
M	64	35	202	266	375	573	246	36714
C*	32	11						
M&C	4	597	3480	4760	6550	9750	3127	38245
M&C	16	150	535	625	775	1000	389	16004
M&C*	64	1170						

Table 5. DECEPTION(3,10);* stagnated far from optimum

We observe a new behavior. Mutation clearly outperforms uniform crossover. But note that a popsize of $N = 16$ is twice as efficient as a popsize of $N = 4$. The performance decreases till $N = 1$. Mutation is most efficient with a popsize between 12 and 24. In very difficult fitness landscapes it pays off to try many different searches in parallel. The BGA with crossover only does not come near to the optimum. Furthermore, increasing the size of the population from 32 to 4000 gives worse result. This behavior of crossover dominates also the BGA with mutation and crossover. The BGA does not find the optimum if it is run with popsizes greater than 50. This is a very unpleasant fact. There exist only a small range of popsizes where the BGA will find the optimum.

It is known that the above problem would vanish, if we use 1-point crossover instead of uniform crossover. But then the results depend on the bit positions of the deceptive function. For the *ugly* deceptive function [21] 1-point crossover performs worse than uniform crossover. Therefore we will not discuss experiments with 1-point crossover here.

The results for SYMBASIN are different. In table 6 the results are given. For mutation this function is only slightly easier to optimize than the DECEPTION function. Good results are achieved with popsizes between 8 and 64. But the SYMBASIN function is a lot more easier to optimize for uniform crossover. The BGA with mutation and crossover performs best. Increasing the popsize decreases the number of generations needed to find the optimum.

OP	N	283	291	294	297	300	SD	FE
M	4	41	1092	2150	3585	7404	4200	29621
M	16	24	125	205	391	765	530	12250
M	64	18	46	68	106	221	136	14172
C*	512	6	16	18	19	20		
C	2048	4	14	15	17	18	0.2	36741
M&C	4	33	1642	2987	5537	9105	1183	36421
M&C	16	15	95	186	331	615	418	9840
M&C	64	12	33	53	90	161	158	10307

Table 6. SYMBASIN(3,10);C*: only 50% reached the optimum

The absolute performance of the BGA is impressive compared to other algorithms. We will only mention ONEMAX and DECEPTION. For ONEMAX the number of function evaluations needed to locate the optimum (FE_{opt}) scales like $e \cdot n \cdot \ln(n)$ (empirical law 1). Goldberg [15] observed a scaling of $O(n^{1.7})$ for his best algorithm. To our knowledge the previous best results for DECEPTION and uniform crossover have been achieved by the CHC algorithm of Eshelman [10]. The CHC algorithm needed 20960 function evaluations to find the optimum. The BGA needs about 16000 function evaluations. The efficiency can be increased if steepest ascent hillclimbing is used [21].

In the last table we will show that the combination of mutation and crossover gives also good results for continuous functions. In table 7 results for Rastrigin's function [22] are shown.

The results are similar to the results of the ONEMAX function. The reason of this behavior has been explained in [22]. A BGA using mutation and discrete recombination with a popsize of $N = 4$ performs most efficiently.

OP	N	1.0	.1	.01	.001	SD	FE
M	4	594	636	691	801	40	3205
M	64	139	176	225	286	9	18316
M&C	4	531	599	634	720	38	2881
M&C	64	50	66	91	123	3	7932

Table 7. Rastrigin’s function ($n = 10$)

10 Conclusion

The theoretical analysis of evolutionary algorithms has suffered in the past from the fact that the methods developed in quantitative genetics to understand especially artificial selection have been largely neglected. Many researchers still believe that the schema theorem [14] is the foundation of the theory. But the schema theorem is nothing else than a simple version of Fisher’s fundamental theorem of natural selection. In population genetics it was discovered very early that this theorem has very limited applications.

We have shown in this paper that the behaviour of evolutionary algorithms can be well understood by the *response to selection* equation. It turned out that the behaviour of the breeder genetic algorithm is already complex for one of the most simple optimization functions, the *ONEMAX* function. This function can play the same role for evolutionary algorithms as the *ideal gas* in thermodynamics. For the ideal gas the thermodynamic laws can be theoretically derived. The laws for real gases are extensions of the basic laws. In the same manner the equations derived for *ONEMAX* will be extended for other optimization functions. For this extension a statistical approach using the concept heritability and the genotypic and phenotypic variance of the population can be used. This approach is already used in the science of artificial breeding.

References

1. H. Asoh and H. Mühlenbein. On the mean convergence time of genetic populations without selection. Technical report, GMD, Sankt Augustin, 1994.
2. Thomas Bäck. Optimal mutation rates in genetic search. In S. Forrest, editor, *5rd Int. Conf. on Genetic Algorithms*, pages 2–9, San Mateo, 1993. Morgan Kaufmann.
3. 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.
4. Thomas Bäck and Hans-Paul Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary Computation*, 1:1–24, 1993.
5. R. K. Belew and L. Booker, editors. *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, 1991. Morgan Kaufmann.
6. H.J. Bremermann, M. Rogson, and S. Salaff. Global properties of evolution processes. In H.H. Pattee, editor, *Natural Automata and Useful Simulations*, pages 3–42, 1966.

7. M. G. Bulmer. *"The Mathematical Theory of Quantitative Genetics"*. Clarendon Press, Oxford, 1980.
8. J. F. Crow. *Basic Concepts in Population, Quantitative and Evolutionary Genetics*. Freeman, New York, 1986.
9. J. F. Crow and M. Kimura. *An Introduction to Population Genetics Theory*. Harper and Row, New York, 1970.
10. L.J. Eshelman. The CHC Adaptive Search Algorithm: How to Have Safe Search when Engaging in Nontraditional Genetic Recombination. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 265–283, San Mateo, 1991. Morgan-Kaufman.
11. D. S. Falconer. *Introduction to Quantitative Genetics*. Longman, London, 1981.
12. R. A. Fisher. *The Genetical Theory of Natural Selection*. Dover, New York, 1958.
13. S. Forrest, editor. *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, 1993. Morgan Kaufmann.
14. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, 1989.
15. D.E. Goldberg. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.
16. D.E. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms revisited: Studies in mixed size and scale. *Complex Systems*, 4:415–444, 1990.
17. Michael Herdy. Reproductive Isolation as Strategy Parameter in Hierarchical Organized Evolution Strategies. In *PPSN 2 Bruxelles*, pages 207–217, September 1992.
18. J.H. Holland. *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press, Ann Arbor, 1975.
19. M. Kimura. *The neutral theory of molecular evolution*. Cambridge University Press, Cambridge University Press, 1983.
20. H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution Algorithms in Combinatorial Optimization. *Parallel Computing*, 7:65–85, 1988.
21. Heinz Mühlenbein. Evolution in time and space - the parallel genetic algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337, San Mateo, 1991. Morgan-Kaufman.
22. Heinz Mühlenbein and Dirk Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm: Continuous Parameter Optimization. *Evolutionary Computation*, 1(1):25–49, 1993.
23. 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.
24. Ingo Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Information*. Fromman Verlag, Freiburg, 1973.
25. H. Schaffer, editor. *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, 1989. Morgan Kaufmann.
26. J.D. Schaffer and L.J. Eshelman. On crossover as an evolutionary viable strategy. In R. K. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68, San Mateo, 1991. Morgan Kaufmann.
27. H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, 1981.
28. G. Syswerda. Uniform crossover in genetic algorithms. In H. Schaffer, editor, *3rd Int. Conf. on Genetic Algorithms*, pages 2–9, San Mateo, 1989. Morgan Kaufmann.