6
Genetic Algorithms

Heinz Mühlenbein
GMD Schloss Birlinghoven
D-53754 Sankt Augustin

1 INTRODUCTION

Evolutionary algorithms which model natural evolution processes were already proposed for optimization in the 60's. I cite just one representative example, the outstanding work of Bremermann [Bremermann, Rogson & Salaff, 1966]. He wrote:

"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 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. I object popular arguments along the lines: "This algorithm is a good optimization method because the principle is used in nature", and vice versa: "This algorithm cannot be a good optimization procedure because you do not find it in nature".

Modelling the natural evolution process and applying it to optimization problems is a challenging task. In principle artificial selection of animals for breeding and selection of virtual animals on a computer is a similar problem. Therefore the designer of an evolutionary algorithm can profit from the knowledge accumulated by human breeders. Later in the course of applying the algorithm to difficult fitness landscapes, the human breeder may also profit from the experience gained by using the algorithm.

Bremermann further notes: "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 to create an offspring string. 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 GA of Holland [1975] and the evolution strategies of Rechenberg [1973] and Schwefel [1981] . 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 mutation in very small populations for continuous parameter optimization.

Both algorithms have been used by small groups for quite a time. Both groups developed a unique notation using a mixture of biological and computer science terms. In this chapter evolutionary algorithms are considered to be part of mathematical random search methods. But the theoretical analysis will be unusual from a mathematical point of view. The analysis is based on classical population genetics and statistics. It provides some answers to the following questions:

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

This approach is opposite to the standard GA analysis initiated by Holland, which starts with the schema theorem [Holland, 1975]. The theorem predicts the progress of schemata under a specific selection schedule, called proportionate selection. Later mutation and recombination are introduced as disruptions of the population. In contrast to this view, I 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. 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. Recombination needs a large population size. In principle, evolutionary algorithms using selection and mutation or selection and recombination alone are able to locate the optimum. Therefore the most difficult question of population genetics and evolutionary algorithms remains: Why using both search strategies together?

In the analysis I distinguish between empirical laws and theorems. Empirical laws are derived from carefully performed computer experiments. Theorems are obtained by purely mathematical reasoning. Empirical laws are by no means less true than theorems. They are laws explaining the results of numerical experiments. This procedure was and is successfully used in physics. A historical example are the laws describing the movements of the planets. Kepler derived his famous laws empirically. They explained all the available data, in addition they could be used for prediction. Newton was able to derive the same laws by postulating a gravitational force between the sun and the planets. Thus in Newton's theory Kepler's laws can be proven mathematically. In my terminology Newton converted an empirical law to a theorem by a theory. The main source of the confusion today is that the word "empirical" has one sense in which it refers to something based purely on observation without theoretical depth. But I use the word in its classical sense. With computers widespread available the empirical approach is a viable alternative to the analytical approach.

The outline of this chapter is as follows. First some of the most popular evolutionary algorithms are broadly described. These algorithms differ on their emphasis on selection, mutation and recombination. In order to evaluate the different algorithms, a theoretical understanding of the algorithms is necessary. Unfortunately, it turns out that such an analysis is very difficult, even in the case of simple functions. In section 3 I will shortly describe the early approaches to understand genetic algorithms. Then a very short survey of population genetics is done. Our analysis starts with the equation for the *response to selection* which predicts the expected average progress of the population. The equation is used to analyze the dynamic behavior of a genetic population with recombination and selection. In section 7 the two loci case is analysed in detail. This analysis leads to a new, more efficient recombination scheme, called *gene pool recombination,* which is discussed in section 8. Selection and mutation are investigated in section 9. After a summary of the major theoretical results, the search strategies of mutation and recombination are compared in section 11. Problems of population genetics had major influence on the development of modern statistics. One problem, estimating the heritability, is described in section 12. The last section shows how to numerically apply the theory.

The theory which will be presented was developed together with a number of researchers. In the following sections I will use we to acknowledge the common work.

2 EVOLUTIONARY ALGORITHMS

A previous survey of search strategies based on evolution has been done by Mühlenbein, Gorges-Schleuter & Krämer [1988]. Evolutionary algorithms for continuous parameter optimization are surveyed in Bäck & Schwefel [1993].

Algorithms which are driven mainly by mutation and selection have been developed by Rechenberg [1973] and Schwefel [1981] for continuous parameter optimization. Their algorithms are called evolution strategies ES. The following strategy is the most famous one.

$(\mu + \lambda)$ Evolution Strategy

STEP1: Create an initial population of size λ.

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

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 (GAs). Sexual reproduction is characterized by recombining two parent strings into an offspring. This recombination is often called *crossover*. Genetic algorithms were invented by Holland [1975]. Recent surveys can be found in Goldberg [1989] and the proceedings of the international conferences on genetic algorithms ICGA [1989],[1991],[1993].

Genetic Algorithm

STEP0: Define a genetic representation of the problem.

STEP1: Create an initial population $P(0) = x_1^0, \ldots, x_N^0$. Set $t = 0$.

STEP2: Compute the average fitness $\bar{f}(t) = \sum_i^N f(x_i)/N$. Assign each individual the normalized fitness value $f(x_i)/\bar{f}(t)$.

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 of the selected parents.

STEP4: Pair all parents at random forming $N/2$ pairs. Apply crossover with a certain probability 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 m each bit of the selected string. The *crossover* operator works with two strings. If two strings $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are given, then the *uniform crossover* operator [Syswerda, 1989] combines the two strings as follows

$$z = (z_1, \ldots, z_n) \;\; z_i = x_i \; or \; z_i = y_i$$

Normally x_i or y_i are chosen with equal probability. In genetic algorithms many different crossover operators are used. We will only consider uniform crossover. In general we will use the more general term *recombination* for any method 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** [Mühlenbein, 1991], 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 using genetic operators like recombination and mutation.

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 *distributed breeder genetic algorithm* **DBGA** [Mühlenbein & Schlierkamp-Voosen, 1993] 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 $P(0)$ of size N. Set $t = 0$.

STEP2: Each individual may perform local hill-climbing.

STEP3: The breeder selects $T\%$ of the population for mating. This gives the selected parents.

STEP4: Pair the parents 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.

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 breeder genetic algorithm uses genetic representations which depend on the problem. Continuous functions are represented by continuous genes, discrete functions by discrete genes. The genetic operators mutation and recombination are different for the two representations. The theory outlined in this chapter can be used for both representations, but we focus on discrete functions.

3 UNDERSTANDING GENETIC ALGORITHMS

Previous studies on why and how the genetic algorithm works have proceeded along two different lines. In the first line of approach, the theoretical one, most arguments are based on the *schema theorem* [Goldberg, 1989]. Mühlenbein [1991] mentioned that this theorem cannot be used to explain the search strategy of the genetic algorithm. The reason is that the schema theorem does not include the gene frequencies of the actual population. Furthermore it describes only the outcome of proportionate selection. Mutation and recombination are only introduced as perturbations.

We do not want to discuss all the wrong interpretations of this theorem, but only the most simple central failure. For simplicity we assume binary functions. Then a schema is defined over the ternary alphabet $\{0, 1, *\}$. A schema matches a particular string if at every location in the schema a 1 matches a 1 in the string, a 0 matches a 0, and a * matches either. Let $m(H, t)$ be the number of occurrences of schema H in the population at generation t. Then the schema theorem computes $m(H, t+1)$ for proportionate selection [Goldberg, 1989] as

$$m(H, t+1) = m(H, t)\frac{f(H, t)}{\bar{f}(t)}. \tag{1}$$

The above equation immediately follows from the definition of proportionate selection. $f(H, t)$ is the average fitness of schema H at generation t, $\bar{f}(t)$ is the average fitness of the population. Both variables depend on the population at generation t. They can only be computed if the population at generation t is known. In order to circumvent this problem, Goldberg [1989,pp.30] argues as follows: "Suppose that a particular schema remains above average an amount $c\bar{f}$ with c a constant. Then

$$m(H, t+1) = m(H, t) \cdot (1 + c).$$

Starting at $t = 0$ and assuming a stationary value of c, we obtain the equation

$$m(H, t) = m(H, 0) \cdot (1 + c)^t.$$

The effect of reproduction is now quantitatively clear; reproduction allocates exponentially increasing numbers of trials to above-average schemata." This argument is used to claim that proportionate selection is allocating the trials in an optimal manner. The failure of this argument is simple: There can't be a schema remaining a constant c above average. The average fitness of the population is steadily increasing, and convergence of the algorithms means that there is only one genotype left in the population.

Because the schema theorem could not be used to predict the behavior of practical genetic algorithms, a second line of research has been tried. This line was experimental and top-down. A full-blown genetic algorithm was run and the parameters of the GA were optimized for a suite of test functions. An example is the work of Grefenstette [1986]. His research

was extended by Schaffer et al. [1989]. Empirically the researchers found that the following formula gave an almost optimal parameter setting for their test suite

$$\ln N + 0.93 \ln m + 0.45 \ln n = 0.56, \tag{2}$$

where

$N :=$ size of the population,
$m :=$ mutation rate,
$n :=$ length of the chromosome.

This equation can be approximated by

$$N \cdot m \cdot \sqrt{n} = 1.7. \tag{3}$$

The formula indicates that an optimal mutation rate should decrease with the size of the population. We will show that this is not the case. The failure of the statistical approach was to extend the regression results beyond the problem domain considered.

The major mistake of the early theoretical research was to underestimate the difficulty of the problem. Surprisingly the researchers did not seriously consider the theory which has been developed in a different field for understanding the evolution of genetic populations — population genetics.

4 A SURVEY OF POPULATION GENETICS

Population genetics tries to analyze the evolution of genetic populations. A genetic algorithm consists of an artificial genetic population. Therefore population genetics should also be applicable to genetic algorithms. Why has the truth of this statement been recognized so lately?

There are two major reasons. The first one was already mentioned in the previous section. GA researchers believed in a new theory based on the schema theorem. The second reason can be found within population genetics itself. Population genetics consists of a diversity of different models and methods, which are not easily understood or transferred to the needs of genetic algorithms. The major obstacle lies in the fact that the theoretical analysis of evolution centered 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, we 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 quantitative genetics. 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. It is interesting to note that *proportionate selection* as used by the simple genetic algorithm is used in population genetics as a model for natural selection.

Unfortunately the behavior of genetic population is very difficult to analyze mathematically. Therefore population genetics developed a set of models and a set of approaches which investigate specific aspects of genetic populations. Historically three different approaches have been tried:

- the phenotypic approach by the biometricians (Galton, Pearson)
- the genotypic approach by the Mendelians
- the statistical approach used by breeders

The biometricians used mainly the concept of *correlation* and *regression* to quantify the relation between offspring and parent. Their analysis centers on quantitative traits. The Mendelians use Mendel's *genetic chance model* to compute the change of the gene frequencies in the population. Mendel's model is restricted to discrete genes. The scientific way of breeding starts with the equation for the *response to selection*. It tries to predict the outcome of selection experiments. Modern textbooks about population genetics describe the Mendelian approach mainly.

For the theory of genetic algorithms all three approaches are useful. There are at least five parameters necessary to describe the initial state and the evolution of an artificial population of a genetic algorithm. They are:

- the population size N
- the initial frequency of the alleles p_0
- the number of loci n
- the mutation rate m
- the intensity of selection I

It would be futile to investigate the genetic algorithm with all five parameters variable. Therefore we will investigate simpler models with one or more parameters fixed. A similar approach has been used in population genetics.

We just summarize the most important models of population genetics. The interested reader should consult Crow and Kimura [1971] or Naglyaki [1992] for a more comprehensive survey.

1. The Fisher-Wright model
This discrete stochastic model is based on Mendel's genetic chance model. It was analyzed by a Markov chain approach. Some analytical results have been obtained for one locus and proportionate selection.

2. The Kimura (diffusion) approach
This continuous time model is a diffusion approximation of the Fisher-Wright model. It was used to obtain additional analytical results for the Fisher-Wright model. Kimura extended it to analyze evolution without selection (genetic drift).

3. Two loci
Researchers in population genetics have not been able to compute analytical results for the Fisher-Wright model for more than one locus with Mendelian recombination. Therefor a specific model was defined. It assumes an infinite population, the evolution of the genotype frequencies is described by deterministic difference equations.

4. Many loci (quantitative genetics)
This model has been developed by the biometricians. It is purely descriptive, no genetics is involved. It is purely phenotypic. Deriving the equations of this model from an extension of Mendel's chance model was and still is a challenge to population genetics.

For these models some mathematical results have been obtained — after 100 years of hard research of famous researchers. It is rather unlikely that GA researchers will find mathematical solutions to some of the problems still open. But progress is to be expected by carefully designed simulation experiments. Such a blending of mathematical analysis and simulation experiments will be mainly done in this chapter.

5 ARTIFICIAL SELECTION

In this section we will first investigate artificial selection by methods described in Falconer [1981]. Later natural selection which is modelled by proportionate selection will be analyzed by the same method. 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 between the population mean fitness of generation $t+1$ and the population mean of generation t. $R(t)$ measures the expected progress of the population.

$$R(t) = \bar{f}(t+1) - \bar{f}(t). \tag{4}$$

Breeders measure the selection with the *selection differential*, which is symbolized by S. $S(t)$ is defined as the difference between the mean fitness of the selected parents $\bar{f}_s(t)$ and the mean fitness of the population

$$S(t) = \bar{f}_s(t) - \bar{f}(t). \tag{5}$$

Figure 1: Regression of truncation selection

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

The prediction of the response to selection starts with

$$R(t) = b(t) \cdot S(t). \tag{6}$$

$b(t)$ is called the *realized heritability.* 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

$$R(t) = b \cdot S(t). \tag{7}$$

This statistical relation involves no genetics; indeed, it goes back to a series of papers written by Pearson in the 1890's. 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_s for s generations of his breeding scheme

$$R_s = \sum_{t=0}^{s} R(t). \tag{8}$$

In order to compute R_s a second equation is needed. In quantitative genetics, several approximate equations for $S(t)$ are proposed [Bulmer, 1980], [Falconer, 1981]. Unfortunately

Trunc	80 %	50 %	40 %	20 %	10 %	1 %
I	0.34	0.8	0.97	1.4	1.76	2.66

Table 1: Selection intensity.

most of 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.

Breeders have found that the dimensionless quantity

$$I = S(t)/\sigma(t) \tag{9}$$

provides a more convenient measure of the strength of selection. $\sigma(t)$ is the standard deviation of the fitness of the individuals, I is called the *selection intensity*. If the fitness values are normally distributed, then I can be computed analytically. A derivation can be found in [Bulmer, 1980]. Simulations have shown that equation (9) is approximately valid for many practical applications, also in case of fitness values which are not normally distributed. In fact, for arbitrary distributions the following estimate has been proven in [Nagaraja, 1981]

$$S(t)/\sigma(t) \leq \sqrt{(100 - Trunc)/Trunc.} \tag{10}$$

In table 1 the relation between the truncation threshold $Trunc$ and the selection intensity I is shown for a normal distribution. A decrease from 50 % to 1 % leads to an increase of the selection intensity from 0.8 to 2.66.

If we insert (9) into (7) we obtain the equation for the *response to selection* [Falconer ,1981].

$$R(t) = b \cdot I \cdot \sigma(t). \tag{11}$$

The science of artificial selection consists of estimating b and $\sigma(t)$. These estimates depend on the specific traits to be improved. The designer of a genetic algorithm has more freedom. In order to maximize the response he can look for a recombination operator operator which *maximizes the product of the realized heritability* and the *standard deviation* of the offspring generation. This design goal is difficult to fulfill. An application of this design principle to continuous fitness function can be found in [Voigt et al., 1995]. There a fuzzy recombination operator is shown to be superior to other recombination operators.

The equation for the response to selection can also be used for analyzing selection methods. Given a certain class of selection methods, all of which have the same selection intensity I, then the preferred selection method is the one generating parent populations with the *highest standard deviation*. Bickle and Thiele [1995] used this method to compare tournament

selection and truncation selection. They showed that tournament selection creates a parent population with a higher standard deviation than truncation selection with the same selection intensity.

We will now apply the response to selection equation to predict the behavior of a genetic algorithm. 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 estimating b is to make a regression of the mid-parent fitness value to the offspring. The mid-parent 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 mid-parent 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 mid-parents. 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 $\sigma(t)$ 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

$$\sigma(t) = \sqrt{n \cdot p(t) \cdot (1 - p(t))}. \tag{12}$$

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 approximately given for the $ONEMAX$ function by*

$$R(t) = \frac{I}{\sqrt{n}} \cdot \sqrt{p(t)(1 - p(t))}. \tag{13}$$

The number of generations needed until equilibrium is approximate

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

$p_0 = p(0)$ 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*

$$p(t+1) - p(t) = \frac{I}{\sqrt{n}} \cdot \sqrt{p(t) \cdot (1 - p(t))}. \tag{15}$$

The difference equation can be approximated by a differential equation

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

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

$$p(t) = 0.5 \left(1 + \sin(\frac{I}{\sqrt{n}} t + \arcsin(2p_0 - 1)) \right). \tag{16}$$

The convergence of the total population is characterized by setting $p(GEN_e) = 1$. *This gives equation (14).*

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

$$\tilde{R}(t) = \frac{\pi}{4.3} \cdot \frac{I}{\sqrt{n}} \cdot \sqrt{p(t)(1 - p(t))} \tag{17}$$

$$\tilde{GEN}_e = \frac{4.3}{\pi} (\frac{\pi}{2} - \arcsin(2p_0 - 1)) \cdot \frac{\sqrt{n}}{I}. \tag{18}$$

These equations fit the results obtained from simulations very good. 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 pop-size* 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. The determination of N^* is outside the scope of this chapter. The interested reader is referred to [Mühlenbein & Schlierkamp-Voosen, 1994b].

6 NATURAL SELECTION

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

$$q_{i,S}(t) = q_i(t) \frac{m_i}{\bar{f}(t),} \tag{19}$$

where $\bar{f}(t) = \sum q_i(t) m_i$ is the average fitness of the population. Note that the above equation is just the schema theorem applied to strings (schemata of length n).

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

$$S(t) = \frac{\sigma^2(t)}{\bar{f}(t).} \tag{20}$$

Let the fitness function be $ONEMAX(n)$. It is assumed that the genotypes are binomial distributed, $\sigma^2(t) \approx np(t)(1-p(t))$. Then the response to selection is given by

$$R(t) = 1 - p(t). \tag{21}$$

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

$$GEN_{1-\epsilon} \approx n \cdot ln\frac{1-p_0}{\epsilon.} \tag{22}$$

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

Proof:

$$\begin{aligned} S(t) &= \sum_{i=1}^{N} q_{i,S} m_i - \bar{f}(t) \\ &= \sum_{i=1}^{N} \frac{q_i(t)m_i^2 - q_i(t)\bar{f}^2(t)}{\bar{f}(t)} \\ &= \frac{1}{\bar{f}(t)} \cdot \sum_{i=1}^{n} q_i(t)(m_i - \bar{f}(t))^2 \\ &= \frac{\sigma^2(t)}{\bar{f}(t)} \end{aligned}$$

For $ONEMAX(n)$ we have $R(t) = S(t)$. Because $\bar{f}(t) = np(t)$, equation (21) is obtained. From $R(t) = n(p(t+1) - p(t))$ we get the difference equation

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

This equation has the solution

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

This equation can be simplified to

$$p(t) = 1 - (1 - \frac{1}{n})^t(1 - p_0). \tag{24}$$

By setting $p(GEN_{1-\epsilon}) = 1 - \epsilon$ equation (22) is easily obtained.

Remark: If we assume $R(t) = S(t)$ we obtain from equation (20) a version of Fisher's *fundamental theorem of natural selection* [Fisher, 1958].

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. This effect was

observed early in practical applications of genetic algorithms. Therefore many different extensions to proportionate selection have been invented. The theory presented here shows that the progress of the genetic population only depends on the selection differential and the variance of the selected parents. The specific selection procedure does not matter. Up to now the only alternative to truncation selection is *tournament selection* [Goldberg, 1991] with a reasonable tournament size. In fact, one can show that the tournament size can be mapped onto the selection intensity defined earlier for truncation selection.

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.

7 TWO LOCI

In this section we will derive exact equations for infinite populations. For simplicity we restrict the discussion to two loci and proportionate selection. In this case there are four possible genotypes: $(0,0),(0,1),(1,0)$, and $(1,1))$ which we index by $i = (0,1,2,3)$. We denote their fitness values m_0, m_1, m_2, and m_3) respectively. Let $q_i(t)$ be the frequency of genotype i at generation t. We assume an infinite population and *uniform crossover*. For proportionate selection the exact equations describing the evolution of the frequencies q_i can be derived easily. These equations — known for diploid chromosomes in population genetics [Crow & Kimura, 1970] — assume an infinite population.

$$q_i(t+1) = \frac{m_i}{\bar{f}(t)} q_i(t) + \epsilon_i \frac{D(t)}{2\bar{f}(t)^2} \quad i = 0,1,2,3 \tag{25}$$

with $\epsilon = (-1,1,1,-1$. $\bar{f}(t) = \sum_{i=0}^{3} m_i q_i(t)$ is the average fitness of the population. $D(t)$ defines the deviation from linkage equilibrium

$$D(t) = m_0 m_3 q_0(t) q_3(t) - m_1 m_2 q_1(t) q_2(t). \tag{26}$$

Note that $D(t) = 0$ if $q_0(t)q_3(t) = q_1(t)q_2(t)$ and $m_0 m_3 = m_1 m_2$. The first condition is fulfilled if the genotypes are binomially distributed. This assumption is called the *Hardy-Weinberg equilibrium* in population genetics. The general nonlinear difference equations have not yet been solved analytically, (see the discussion by [Naglyaky, 1992], but it is possible to derive an exact expression for the realized heritability. By summation we obtain

$$R(t) = \bar{f}(t+1) - \bar{f}(t) = \frac{V(t)}{\bar{f}(t)} - (m_0 + m_3 - m_1 - m_2)\frac{D(t)}{2\bar{f}(t)^2,} \tag{27}$$

where $V(t) = \sigma^2(t) = \sum q_i(t)(m_i - \bar{f}(t))^2$ denotes the variance of the population. Using $S(t) = V(t)/\bar{f}(t)$ we obtain the exact equation for the heritability,

$$b(t) = 1 - (m_0 - m_1 - m_2 + m_3)\frac{D(t)}{2\bar{f}(t)V(t)}. \tag{28}$$

In general, $b(t)$ depends on the genotype frequencies. Note that $b(t) = 1$ if $D(t) = 0$ or $m_0 + m_3 = m_1 + m_2$. The second assumption is fulfilled for the function ONEMAX(2) which has the fitness values $m_0 = 0, m_1 = m_2 = 1, m_3 = 2$. From (25) we obtain

$$\begin{aligned} \bar{f}(t+1) &= q_1(t+1) + q_2(t+1) + 2q_3(t+1) \\ &= \frac{q_1(t) + q_2(t) + 4q_3(t)}{\bar{f}(t)} \\ &= 1 + \frac{2q_3(t)}{\bar{f}(t)}. \end{aligned}$$

Let $p(t)$ denote the frequency of allele 1. Then by definition $\bar{f}(t) = 2p(t)$. Therefore we obtain

$$R(t) = 1 - p(t) + \frac{B_3(t)}{p(t)}, \tag{29}$$

where $B_3(t)$ denotes how far $q_3(t)$ deviates from the frequency given by the binomial distribution:

$$B_3(t) = q_3(t) - p^2(t). \tag{30}$$

The exact difference equation for $p(t)$ can be written as

$$p(t+1) = p(t) + \frac{1}{2}(1 - p(t)) + \frac{B_3(t)}{2p(t)}. \tag{31}$$

This equation has two unknown variables, $p(t)$ and $q_3(t)$. Therefore $p(t)$ cannot be directly computed. Selection leads the population away from the binomial distribution, and TPR is not able to recreate a binomial distribution for the offspring population.

We now discuss a function where $D(t) = 0$ if the population starts in a Hardy-Weinberg equilibrium. An example is MULT(2) with fitness values $m_0 = 1, m_1 = m_2 = 2, m_3 = 4$. In this case the difference equation for $p(t)$ is given by

$$p(t+1) = p(t)\frac{2}{1 + p(t)}, \tag{32}$$

which can be solved easily.

In summary, even linear fitness functions lead to difficult systems of difference equations. The genetic population moves away from Hardy-Weinberg equilibrium. A class of multiplicative fitness functions with $m_0 m_3 = m_1 m_2$ leads to simpler equations, because the population stays in Hardy-Weinberg equilibrium.

8 GENE POOL RECOMBINATION

The exact analysis of recombination together with selection leads to difficult nonlinear differential equations. Recombination of two genotypes creates a linkage between the genes at

different loci. This linkage is very hard to describe mathematically. Therefore we decided to look for a recombination operator that leads to simpler equations, like those we used as an approximation. This operator must create a binomial distribution. Fortunately, there is a simple recombination scheme that fulfills this condition; we call it *gene pool recombination (GPR)*.

Definition: *In gene pool recombination the two "parent" alleles of an offspring are randomly chosen with replacement from the gene pool given by the parent population* **selected** *before. Then the offspring allele is computed using any of the standard recombination schemes for TPR.*

For binary functions GPR is obviously a Bernoulli process. Let $p_i^s(t)$ be the frequency of allele 1 at locus i in the *selected* parent population. Then GPR creates offspring with allele frequency $p_i(t+1) = p_i^s(t)$ and variance $p_i(t+1)(1-p_i(t+1))$ at locus i.

The idea of using more than two parents for recombination is not new. Already Mühlenbein [1989] used eight parents; the offspring allele was obtained by a majority vote. Multi-parent recombination has also been investigated recently by Eiben et al. [1994], though their results are somewhat inconclusive. For binary functions the bit-based simulated crossover (BSC) of Syswerda [1993] is similar to GPR. However, his implementation merged selection and recombination. An implementation of BSC which separates selection and recombination was empirically investigated by Eshelman and Schaffer [1993]. GPR is an extension of BSC, it can be used for any representation — discrete or continuous. Variants of GPR have been also successfully used for the traveling salesman problem (see the discussion by Johnson in this book).

In order to analyze GPR we will derive difference equations for the gene frequencies, valid for arbitrary fitness functions and infinite populations. As before, we restrict the analysis to the case of two loci and proportionate selection.

Let $q_i(t)$ be the frequency of genotype i at generation t. For$n = 2$ loci, the marginal gene frequencies $p_1(t)$ and $p_2(t)$ can be obtained from

$$\begin{aligned} p_1(t) &= q_2(t) + q_3(t) \\ p_2(t) &= q_1(t) + q_3(t). \end{aligned}$$

We assume that the initial population has a binomial distribution. This means that

$$\begin{aligned} q_0(0) &= (1-p_1(0))(1-p_2(0)) \\ q_1(0) &= (1-p_1(0))p_2(0) \\ q_2(0) &= p_1(0)(1-p_2(0)) \\ q_3(0) &= p_1(0)p_2(0). \end{aligned}$$

Then the following theorem holds:

Theorem 3 *Let the initial population have a binomial distribution. For an infinite population with GPR and proportionate selection, the marginal frequencies $p_1(t)$ and $p_2(t)$ can be*

obtained from

$$p_1(t+1) = p_1(t)\frac{m_2(1-p_2(t)) + m_3 p_2(t)}{\bar{f}(t)} \tag{33}$$

$$p_2(t+1) = p_2(t)\frac{m_1(1-p_1(t)) + m_3 p_1(t)}{\bar{f}(t)}. \tag{34}$$

The realized heritability, $b(t)$, is given by

$$b(t) = 1 - (m_0 m_3 - m_1 m_2)(m_0 - m_1 - m_2 + m_3)\,\frac{p_1 p_2 (1-p_1)(1-p_2)}{V\bar{f}} \tag{35}$$

where p_1, p_2, V, and $\bar{f}$ depend on t.

Proof: *Proportionate selection selects the genotypes for the parents of population $t+1$ according to*

$$q_i^s(t) = \frac{m_i}{\bar{f}(t)} q_i(t).$$

From q_i^s the marginal frequencies p_1^s and p_2^s can be obtained from the two equations $p_1^s(t) = q_2^s(t) + q_3^s(t)$ and $p_2^s(t) = q_1^s(t) + q_3^s(t)$. For each locus, GPR is a Bernoulli process; therefore, the marginal gene frequencies of parents and offspring remain constant

$$p_1(t+1) = p_1^s(t) \quad p_2(t+1) = p_2^s(t).$$

Combining these equations gives equations 33 and 34. The expression for the realized heritability can be obtained after some manipulations.

Remark: Theorem 3 can be extended to arbitrary functions of size n, or genetically speaking to n loci. This means that the evolution of an infinite genetic population with GPR and proportionate selection is fully described by n equations for the marginal gene frequencies. In contrast, for TPR one needs 2^n equations for the genotypic frequencies. For GPR one can in principle solve the difference equations for the marginal gene frequencies instead of running a genetic algorithm.

Note that $b(t) = 1$ if $m_0 m_3 = m_1 m_2$ or if $m_1 + m_2 = m_0 + m_3$. Let us first consider ONEMAX(2). The average fitness is given by $\bar{f}(t) = p_1(t) + p_2(t)$. If $p_1(0) = p_2(0)$, we have $p_1(t) = p_2(t) = p(t)$ for all t. From $\bar{f}(t) = 2p(t)$ we obtain

$$R(t) = 1 - p(t) \tag{36}$$

and

$$p(t+1) = p(t) + \frac{1}{2}(1 - p(t)). \tag{37}$$

This equation is similar to the equation obtained for TPR. It has be solved in the previous section. Both equations become equal if $B_3(t) = 0$. This shows that for linear fitness

functions, GPR and TPR give similar results — with a slight advantage for GPR, which converges faster.

Let us now turn to the function MULT(2). Combining $\bar{f}(t) = (1+p(t))^2$ with equation (33), we obtain equation (32). For MULT(2) TPR and GPR lead to the same difference equation. One can show that in general for multiplicative functions ($m_0 m_3 = m_1 m_2$) TPR and GPR give the same gene frequencies.

For many loci the above analysis can easily be extended to fitness functions which are called "unitation" functions. For these functions the fitness values depend only on the number of $1's$ in the genotype. Again for simplicity we consider three loci only. Let u_i denote the fitness of a genotype with i $1's$. Under the assumption that $p_1(0) = p_2(0) = p_3(0)$, all marginal frequencies have the same value, which we denote as $p(t)$. Then we obtain for the marginal frequency $p(t+1) = c \cdot p(t)$

$$c = \frac{u_1(1-p)^2 + 2u_2 p(1-p) + u_3 p^2}{u_0(1-p)^3 + 3u_1 p(1-p)^2 + 3u_2 p^2(1-p) + u_3 p^3}, \tag{38}$$

where $p = p(t)$. If $c > 1$ the marginal frequency $p(t)$ increases, if $c < 1$ it decreases. As a specific example we analyze a "deceptive" function of 3 loci, see Goldberg [1989]. Let the fitness values of this function be $u_0 = 28, u_1 = 26, u_3 = 30$. The global optimum is at 111, the local optimum at 000. The fitness value for u_2 can be varied. Depending on u_2 and the initial population, the genetic population will converge to 000 or 111. Take $u_2 = 0$ as example. If $p_0 > 0.639$ the population will converge to 111. If $p_0 < 0.639$ the population will converge to 000. At $p_0 = 0.639$ we have $c = 1$. This point is an *unstable equilibrium point.*

Remark: The analysis of unitation functions of three or more loci shows that a genetic algorithm using selection and recombination only is **not a global optimization** method. Depending on the frequency distribution of the genotypes and the fitness values, a genetic algorithm with infinite population size will deterministically converge to one of the local optima. The equations derived in this chapter can be used to determine the optima to which the genetic algorithm will converge.

As a last example we will analyze a fitness function where the results of TPR and GPR are very different. For simplicity we take two loci. The fitness values are defined as $m_0 = 0.99, m_1 = 0, m_2 = 0$, and $m_3 = 1$.

Table 2 shows that GPR very slowly changes the gene frequencies at the beginning. In fact, if $m_0 = 1$ the population would stay in equilibrium. After three generations GPR changes the gene frequencies very quickly. In contrast, TPR dramatically changes the frequencies in the first generation. The population immediately goes to the equilibrium points for the symmetric fitness function $m_0 = m_3 = 1, m_1 = m_2 = 0$. It takes TPR a long time to leave this equilibrium point and march to the optimum.

Proportionate selection should not be used for a genetic algorithm, its selection intensity is far too low. Unfortunately, for other selection methods the equations for the marginal gene frequencies are difficult to obtain. For truncation selection an approximate analysis can

t	REC	q_0	q_1	q_3	$\overline{f}$	Var
0	TPR	0.250	0.250	0.250	0.4975	0.2475
1	TPR	0.372	0.125	0.378	0.7463	0.1857
2	TPR	0.369	0.125	0.381	0.7463	0.1857
3	TPR	0.365	0.125	0.385	0.7464	0.1856
9	TPR	0.287	0.121	0.471	0.7556	0.1819
0	GPR	0.250	0.250	0.250	0.4975	0.2475
1	GPR	0.247	0.250	0.252	0.4975	0.2475
2	GPR	0.242	0.250	0.257	0.4977	0.2476
3	GPR	0.233	0.250	0.268	0.4983	0.2477
6	GPR	0.120	0.226	0.427	0.5457	0.2467
9	GPR	0.000	0.006	0.988	0.9883	0.0115

Table 2: Results for TPR and GPR for a bimodal function

be done by using the equation for the response to selection. Here one has to estimate the standard deviation of the phenotypes. This was already done for the ONEMAX function in theorem 1. This theorem is exaxt for GPR, because GPR leads to a binomial distribution. Our analysis shows that GPR converges for the ONEMAX function about 25% faster than TPR.

9 SELECTION AND MUTATION

The analysis of mutation is divided into two parts. First very small populations are considered. Then mutation in large populations is investigated.

9.1 Mutation in small populations

The analysis of the mutation operator will first be done for an evolutionary algorithm with just one parent and one offspring. This algorithm can be called a random walk or *iterated hill-climbing* if a hill-climbing strategy is used in addition.

Iterated hill-climbing (Random_walk algorithm)

STEP1: Generate a random initial string s.

STEP2: If $hc \neq \emptyset$, do the hill-climbing strategy hc, giving a modified s'.
If $f(s') \geq f(s)$ then $s := s'$.

STEP3: Mutate each bit of s with probability m.

STEP4: If not finished, return to STEP2.

The above algorithm is one of the simplest optimization methods. It will locate with a probability greater zero the global optima. The algorithm has to be distinguished from a *multi-start* algorithm. In iterated hill-climbing the new configuration is generated by a random perturbation of the previous found local optimum. If this perturbation is too low, then the new configuration will be nearby to the local optimum. If the perturbation is too strong, then the new configuration is far away from the current search. In a multi-start algorithm each initial configuration is randomly generated.

The importance of using a hill-climbing method to speed up the search has been shown in Mühlenbein [1991] and [1992]. There a detailed analysis of the above algorithm was made by a Markov chain analysis. In order to compare mutation with crossover we just cite the following theorem from Mühlenbein [1992].

Theorem 4 *Let the initial string have $n \cdot p_0$ bits correct. Then for $ONEMAX$ of size $n \gg 0$ and a mutation rate of $m = j/n$ with $j \ll n$, the expected number of trials $< T(m) >$ to reach the optimum is approximately given by*

$$< T(m) > \approx e^j \frac{n}{j} \sum_{i=1}^{(1-p_0)n} \frac{1}{i}. \tag{39}$$

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

$$< T(j/n) > \approx e^j \cdot n/j \cdot ln((1-p_0)n). \tag{40}$$

The number of trials T needed to reach the optimum increases exponential in j compared to the optimal mutation rate of $1/n$. A small absolute change of the mutation rate may have a dramatic impact. Therefore we get for a fixed mutation rate the following result.

Corollary: *The optimal mutation rate for unimodal discrete functions is inversely proportional to the size of the chromosome.*

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 et al. [1966].

It has to be mentioned that with a mutation rate of $m = 1/n$ the string will not be changed at all with probability $P = (1 - 1/n)^n \approx e^{-1}$. But if the population consists of one string only, it makes no sense not to change the string. Therefore the mutation rule for small populations is as follows: *Change each bit with probability $m = 1/n$. If the chromosome is not mutated at all, randomly change one bit.*

We have confirmed equation (40) by intensive simulations [Mühlenbein, 1991]. Recently Bäck [1993] has shown that $< T(m) >$ can be only marginally reduced if a theoretically optimal *variable* mutation rate is used. This variable rate depends on the number of bits

yet to be corrected. This result has been predicted in [Mühlenbein, 1992]. Mutation spends most of the time in adjusting the very last bits. But in this region the optimal mutation rate is obviously $m = 1/n$.

Theorem 4 cannot easily be extended to a larger number of offspring. Therefore we will only qualitatively discuss this problem by simulations. Some results are displayed in table 3. One clearly observes the law of diminishing returns. Increasing the population size N reduces $< T(m) >$, the number of generations needed to find the optimum, less and less. Mutation is most efficient with a small number of offspring.

N	2	4	8	16	32	64	128	256	512	1024
$< T(m) >$	750	400	200	115	71	48	37	30	26	23
Sp	2.0	3.75	7.5	13.4	21	31	41	50	58	65

Table 3: $< T >$ and speedup Sp. N offspring from one parent (ONEMAX(128))

The speedup Sp shows how much faster the solution is obtained with a larger number of offspring N. It is defined as the quotient of $< T > (1)/ < T > (N)$. The speedup is almost linear for small N and seems to slow down to a logarithmic function. This indicates that mutation is not an efficient search in a large population. We will show in the next section that smaller selection intensities give still worse results.

9.2 Mutation in large populations

In order to analyze mutation in large populations one has to extend the equation for the response to selection. The concept of heritability is not of much use here. The following theorem is one possibility to extend the concept of the response to selection.

Theorem 5 *Let s_t be the probability of a mutation success, imp the average improvement of a successful mutation. Let f_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*

$$R(t) = S(t) + s_t \cdot imp - f_t \cdot red. \tag{41}$$

Proof: *Let $\bar{f}_s(t)$ be the average of the selected parents. Then*

$$\bar{f}(t+1) = \bar{f}_s(t) + s_t \cdot imp - f_t \cdot red.$$

Subtracting $\bar{f}(t)$ 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 s_t and f_t has been independently discovered by Schaffer et al. [1991]. They did not use the difference of the probabilities, but the quotient which they called the *safety factor*.

$$safety = \frac{s_t}{f_t}$$

The following empirical law has been found by simulations [Mühlenbein & Schlierkamp-Voosen, 1994a].

Empirical Law 1 *For $ONEMAX$ of size n, 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 only is approximate*

$$R(t) = 1 + (1 - p(t))e^{-p(t)} - p(t)e^{-(1-p(t))} \tag{42}$$

Equation (42) 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*

$$R(t) = 2 - 2p(t) \tag{43}$$

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

$$GEN_{1-\epsilon} \approx \frac{n}{2} \cdot ln\frac{1 - p_0}{\epsilon} \tag{44}$$

By comparing theorem 2 with empirical law 2, one obtains that mutation with 50% truncation selection is twice as efficient as proportionate selection with crossover. This shows once more the inefficiency of proportionate selection.

Next we will compare the theoretical results with simulations. In figure 2 the development of the mean fitness is shown. The simulations have been done with two pop-sizes ($N = 1024, 64$) 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 a large population is inefficient for mutation. Furthermore the figure shows that mutation spends most of its time in getting the last bits correct.

A large mutation rate has an interesting effect. The mean fitness increases faster at the beginning, but it never finds the optimum. This might suggest to use a variable mutation

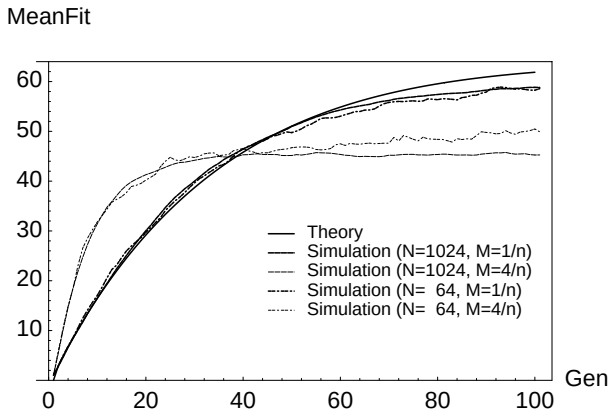

Figure 2: Theory (43) and simulation results for various N and mutation probabilities M

rate. But we have to add a caveat to this idea. We have already mentioned in the previous section that the increase in performance by using a variable mutation rate will be rather small.

The efficiency of mutation in large populations can be increased by very strong selection. It is obvious that for unimodal functions one should only select the best individual as parent.

The major results of this section can be summarized as follows: *The mutation operator in large populations is not effective. It needs very strong selection. Furthermore, the efficiency of the mutation operator critically depends on the mutation rate.*

10 SUMMARY OF MAJOR RESULTS

In the previous sections we have presented the framework for analyzing evolutionary algorithms. We now summarize the results and include additional results which have been proven in [Mühlenbein & Schlierkamp-Voosen, 1993,1994a,1994b]. Most of the results are exact for gene pool recombination and only approximate for two parent mating.

A fascinating phenomenon of genetic populations is called *genetic drift*. Any finite genetic population of size N will converge to a single genotype, even if selection is not applied. The expected number of generations until convergence GEN_e is surprisingly low.

Let n denote the number of genes, N the size of the population. Then the expected number of generations until equilibrium is given by [Asoh & Mühlenbein, 1994a]

$$E(GEN_e) \approx 1.4 \cdot N \cdot (0.5 \ln(n) + 1). \tag{45}$$

The above equation is valid for random mating with recombination but without selection and mutation. Note that the $E(GEN_e)$ scales linearly in N and only logarithmically in n.

Genetic drift is the reason for the surprising fact that *small selection intensities decrease the probability to find the optimum.*

We now turn to truncation selection. If the size N of the population is larger than the *critical pop-size* N^*, the minimum population size to converge to the optimum with high probability, then we have for the expected number of generations until convergence

$$GEN_e \approx \left(\frac{\pi}{2} - \arcsin(2p_0 - 1)\right) \cdot \frac{\sqrt{n}}{I}. \tag{46}$$

Note that GEN_e is independent of N. The estimation of the critical population size N^* is very difficult. We conjectured in [Mühlenbein & Schlierkamp-Voosen, 1994b]

$$N^* \approx \sqrt{n} \cdot \ln(n) \cdot f_1(p_0) \cdot f_2(I). \tag{47}$$

Proportionate selection as used by the simple GA [Goldberg, 1989] selects too weakly when the variance of the population becomes small. The expected number of generations $GEN_{1-1/n}$ until the favorable allele is distributed in the population with probability of $1-1/n$ is given by

$$GEN_{1-1/n} \approx n \cdot \ln\left(n \cdot (1 - p_0)\right). \tag{48}$$

$GEN_{1-1/n}$ is much larger than the corresponding value for truncation selection.

All the above equations are valid for a mutation rate of 0. We now turn to populations using only mutation. The most important result concerns the mutation rate.

Rule of thumb: *The mutation rate $m = 1/n$ where n is the size of the chromosome is almost optimal .*

For the above mutation rate the expected number of generations GEN_{opt} until the optimum is found has been computed for the $(1+1)$−strategy (one parent, one offspring; the better of the two survives).

$$GEN_{opt} \approx e \cdot n \cdot \ln\left(n \cdot (1 - p_0)\right). \tag{49}$$

Mutation in a large population is inefficient. The asymptotic scaling of GEN_{opt} is independent of the pop-size N. It stays at $O(n \cdot \ln(n))$. For very large pop-sizes GEN_{opt} is given by

$$GEN_{1-1/n} \approx \frac{n}{2} \cdot \ln\left(n \cdot (1 - p_0)\right). \tag{50}$$

The above equation is valid for a large population and a truncation selection threshold of $T = 0.5$. Note that the above value is about half the value of proportionate selection.

The above theorems show that for binary representations populations using either recombination or mutation are able to locate the optimum. If $p_0 = 0.5$ i.e. half of the bits are correct in the initial population, the asymptotic order of the number of trials needed (FE_{opt}), seems to be the same, namely $O(n \cdot \ln(n))$. For recombination this number is obtained by multiplying GEN_e with the critical pop-size N^*. Therefore the question which of the two operators is more efficient is difficult to answer. The comparison needs an exact expression for N^*, which has not yet been obtained. But one can easily make a qualitative comparison. The major difference between mutation and recombination is their dependence on p_0, the percentage of the desired allele in the initial population.

Let us take $p_0 = 1 - 1/n$ as example. Here, only one bit is wrong on the average. Mutation will need about $O(n)$ trials to change the incorrect bit. Uniform crossover of two strings, each with one bit wrong, will generate the optimum string with probability $1/4$, independent of the size of the problem. Thus recombination is much more efficient than mutation. But the determination of the exact N^* is also difficult in this simple case. It will need on the average 4 trials to generate the optimum. But the probability that a pop-size of 4 will not generate the optimum is $0.75^4 = 0.31$. It needs 16 trials in order to obtain the optimum with 99% probability.

If we take $p_0 = 1/n$ the situation is reversed. Now mutation is much more efficient than recombination which needs a huge pop-size in order to locate the optimum. Recombination has too few building blocks to generate better offspring.

Most of our analytical results have been derived under the assumption of additive genetic effects. This theory explains the behavior of the most important evolutionary forces. It plays a similar role for the BGA as the "ideal gas" theory for thermodynamics. There exists no *ideal gas* in reality, but the ideal gas theory gives much insight into the overall behavior of gases. In order to understand evolutionary algorithms in more complex fitness landscapes, we have to extend the theory by using more advanced statistical methods. This is the topic of one of the next sections. Before we will discuss how to combine search by mutation and search by recombination.

11 MUTATION AND RECOMBINATION

In the previous sections the search strategies of the recombination and the mutation operator have been individually analyzed. We will now compare them. In figure 3 recombination with truncation selection and with proportionate selection is compared to mutation with truncation selection. The fitness function is $ONEMAX(64)$. The initial population was generated with $p_0 = 1/64$. As predicted by the theory, the mean fitness of the population with mutation is best at the beginning. When $p(t) \geq 0.5$ recombination with truncation selection performs best. The simulations confirm that recombination with proportionate selection is twice as inefficient as mutation with a truncation threshold of 50%.

The question now arises how to best combine mutation and recombination. This can be done

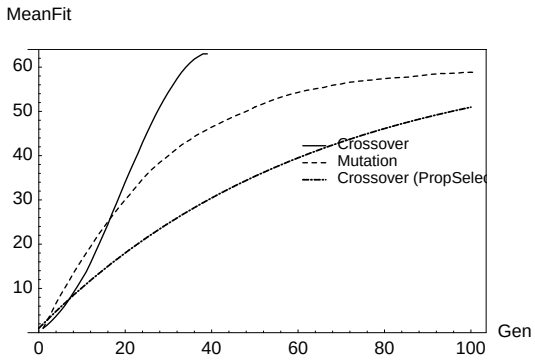

Figure 3: Comparison of crossover and mutation

by two different methods at least. One can try to use both operators in a single evolutionary algorithm with an almost optimal parameter setting. This means that a good mutation rate and a good population size has to be predicted. This method is used in the breeder genetic algorithm. A good mutation rate is constantly applied. The variance of the population remains high enough for recombination to be effective. A different approach is used by Eshelman [1991] in the CHC algorithm. CHC is based on recombination. If the variance of the population is below a certain threshold the population is thoroughly changed by applying mutation vigorously. This event gives recombination the chance for further improvements. The success of this method depends on the right amount of change. If too much is changed then this would be just a new start of the algorithm. If the changes are too small then the population will stay in equilibrium.

We believe that for difficult optimization problems a different method is more suited. If optimal parameter settings or strategies are not known beforehand, one can try a competition between subpopulations using different strategies.

Such a competition can be done on different levels, for example the level of the individuals, the level of subpopulations or the level of populations. For evolution strategies, Bäck, Hofmeister & Schwefel [1991] have implemented the adaptation of strategy parameters on the level of the individual. The strategy parameters of the best individuals are recombined, giving the new step-size for the mutation. Herdy [1992] uses competition on the population level. In this case whole populations are evaluated at certain intervals. The strategies of the successful populations proliferate, strategies in populations with bad performance die out.

The adaptation of the breeder genetic algorithm lies between these two extreme cases. The competition is done between subpopulations. It uses a *quality criterion* to rate the groups, 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 has been described in [Schlierkamp-Voosen & Mühlenbein, 1994].

12 STATISTICS AND GENETICS

In this section we will present two methods for estimating the *heritability*. The first one will use the concept of *regression of offspring to parent* and the second one the concept of decomposing the *genetic variance*. Both methods have been of great importance in the development of statistics and population genetics. In this chapter we will use a slightly different notation which is more common in statistics.

The first theorem connects the realized heritability $b_t = R(t)/S(t)$ with the regression coefficient between *mid-parent* and offspring. Let f_i, f_j be the phenotypic values of parents i and j, then

$$\bar{f}_{i,j} = \frac{f_i + f_j}{2}$$

is called the mid-parent value. Let the stochastic variable $\bar{F}$ denote the mid-parent value.

Theorem 6 *Let $F(t) = (f_1, \ldots, f_N)$ be the fitness values of the population at generation t, where f_i denotes the fitness value of individual i. Assume that an offspring generation $O(t)$ is created by random mating, without selection. If the regression equation*

$$o_{ij}(t) = a(t) + b_{\bar{F}O}(t) \cdot \frac{f_i + f_j}{2} + \epsilon_{ij} \tag{51}$$

with

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

is valid, where o_{ij} is the fitness value of the offspring of i and j, then

$$b_{\bar{F}O}(t) \approx b_t. \tag{52}$$

Proof: *From the regression equation we obtain for the expected averages*

$$E(O(t)) = a(t) + b_{\bar{F}O}(t)\bar{f}(t).$$

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

$$E(O(t)) = \bar{f}(t)$$

Let us now select a subset as parents. The parents will be randomly mated, producing the offspring generation. If the subset is large enough, we may use the regression equation and obtain for the averages

$$\bar{f}(t+1) = a(t) + b_{\bar{F}O}(t) \cdot \bar{f}_s(t).$$

Here $\bar{f}(t+1)$ is the average fitness of the offspring generation produced by the selected parents. Subtracting the above equations we obtain

$$\bar{f}(t+1) - \bar{f}(t) = b_{\bar{F}O}(t) \cdot (\bar{f}_s(t) - \bar{f}(t))$$

This proves $b_{\bar{F}O}(t) = b_t$.

The problem of computing a good regression coefficient is solved by the theorem of Gauss-Markov. We just cite the theorem. The proof can be found in [Rao, 1973].

Theorem 7 *A good estimate for the regression coefficient of mid-parent and offspring is given by*

$$b_{\bar{F}O}(t) = \frac{cov(O(t), \bar{F}(t))}{var(\bar{F}(t))}. \tag{53}$$

The covariance of O and $\bar{F}$ is defined by

$$cov(O(t), \bar{F}(t)) = \frac{1}{N} \sum_{i,j} (o_{i,j} - av(O(t))) \cdot (\bar{f}_{i,j} - av(\bar{F}(t))).$$

av denotes the average and var the variance. Closely related to the regression coefficient is the correlation coefficient $cor(\bar{F}, O)$. It is given by

$$cor(\bar{F}(t), O(t)) = b_{\bar{F}O}(t) \cdot \left(\frac{var(\bar{F}(t))}{var(O(t))} \right)^{1/2}.$$

The above theorem enables us to estimate the heritability by a second method. It works as follows. For a large sample population F offspring are created by random mating. Then the regression coefficient $b_{\bar{F}O}$ is computed by equation (53). This estimate is called *heritability in the narrow sense* [Falconer, 1981]. The relation between realized heritability and heritability in the narrow sense is one of the most difficult parts in population genetics.

The next theorem shows the connection between *mid-parent* and *parent* regression.

Theorem 8 *Mid-parent and parent regression are connected by*

$$b_{FO}(t) = 0.5 \cdot b_{\bar{F}O}(t) \quad cor(F(t), O(t)) = \sqrt{\frac{1}{2}} cor(\bar{F}(t), O(t)). \tag{54}$$

Proof: *We have*

$$cov(O(t), \bar{F}(t)) = cov(O(t), F(t))$$
$$var(\bar{F}(t)) = 0.5 \cdot var(F(t))$$

From (53) the theorem is obtained.

We now describe a method for estimating the covariance. This method connects a microscopic genetic chance model and the macroscopic phenotypic covariance. It is restricted to discrete genes. We only give the necessary definitions and the fundamental theorem. The interested reader is referred to [Asoh & Mühlenbein, 1994a] where the proof can be found.

Let a haploid chromosome with n binary genes x_i be given. Let $\mathbf{x} = (x_1, x_2, \ldots x_n)$ be a genotype, $f(\mathbf{x})$ its fitness. Let the genetic chance model be defined by *uniform crossover*. This model can be considered as Mendel's chance model restricted to haploid chromosomes. We will decompose the fitness value $f(\mathbf{x})$ recursively into an additive part and interaction parts. Let $p(\mathbf{x})$ denote the probability of $\mathbf{x}$, $p(\mathbf{x}|x_i)$ the conditional probability of $\mathbf{x}$ given x_i. First we extract the average.

$$f(\mathbf{x}) = av(f) + r_0(\mathbf{x}) \tag{55}$$

Then we extract the first order (additive) part from the residual $r_0(\mathbf{x})$

$$r_0(\mathbf{x}) = \sum_{i=1}^{n} f_{(i)}(x_i) + r_1(\mathbf{x}), \tag{56}$$

where $f_{(i)}(x_i)$ are given by

$$f_{(i)}(x_i) = \sum_{\mathbf{x}|x_i} p(\mathbf{x}|x_i) r_0(\mathbf{x}) = \sum_{\mathbf{x}|x_i} p(\mathbf{x}|x_i) f(\mathbf{x}) - av(f).$$

Here $\sum_{\mathbf{x}|x_i}$ means that the i-th locus is fixed to the value x_i. The $f_{(i)}(x_i)$ minimize the quadratic error $\sum_{\mathbf{x}} p(\mathbf{x}) r_1(\mathbf{x})^2$.

If $r_1(\mathbf{x}) \not\equiv 0$, we can proceed further to extract the second order terms from $r_1(\mathbf{x})$:

$$r_1(\mathbf{x}) = \sum_{\substack{(i,j) \\ i<j}} f_{(i,j)}(x_i, x_j) + r_2(\mathbf{x}), \tag{57}$$

where

$$\begin{aligned} f_{(i,j)}(x_i, x_j) &= \sum_{\mathbf{x}|x_i,x_j} p(\mathbf{x}|x_i, x_j)\; r_1(\mathbf{x}) \\ &= \sum_{\mathbf{x}|x_i,x_j} p(\mathbf{x}|x_i, x_j)\; f(\mathbf{x}) - f_{(i)}(x_i) - f_{(j)}(x_j) \end{aligned}$$

If we have n loci, we can iterate this procedure $n-1$ times recursively and finally we get the decomposition of f as

$$\begin{aligned} f(\mathbf{x}) &= \bar{f} + \sum_i f_{(i)}(x_i) + \sum_{(i,j)} f_{(i,j)}(x_i, x_j) + \cdots \\ &+ \sum_{\substack{(i_1,\ldots,i_{n-1}) \\ i_1<\ldots<i_{n-1}}} f_{(i_1,\ldots,i_{n-1})}(x_{i_1}, \ldots, x_{i_{n-1}}) + r_{n-1}(\mathbf{x}). \end{aligned}$$

Let V_k for $k = 1$ to $n-1$ be defined as

$$V_k = \sum_{\substack{(i_i,\ldots,i_k) \\ i_1<\ldots<i_k}} \sum_{x_{i_1},\ldots,x_{i_k}} p(x_{i_1},\ldots,x_{i_k}) f_{(i_i,\ldots,i_k)}(x_{i_i},\ldots,x_{i_k})^2, \tag{58}$$

and

$$V_n = \sum_{\mathbf{x}} p(\mathbf{x}) r_{n-1}(\mathbf{x})^2. \tag{59}$$

Now the fundamental theorem of classical population genetics can be formulated.

Theorem 9 *Let the population be in linkage equilibrium i.e.*

$$p(\mathbf{x}) = \prod_{i=1}^{n} p_i(x_i). \tag{60}$$

Then the variance of the population is given by

$$var(F) = V_1 + V_2 + \cdots + V_{n-1} + V_n. \tag{61}$$

The covariance of mid-parent and offspring can be computed from

$$cov(\bar{F}, O) = \frac{1}{2}V_1 + \frac{1}{4}V_2 + \cdots + \frac{1}{2^n}V_n = \sum_{k=1}^{n} \frac{1}{2^k} V_k. \tag{62}$$

From theorems 7 and 9 we obtain

Corollary 1 *If the fitness function is additive, that is $f(\mathbf{x}) = \sum_i f_i(x_i)$, then*

$$cor(\bar{F}, O) = \sqrt{1/2} \qquad b_{\bar{F}O} = 1 \tag{63}$$

The above theorem plays an important role in the science of breeding. Breeders conjecture that the *additive genetic variance* V_1 is the most important factor of the heritability in the narrow sense. The higher order interactions contribute much less to the heritability. Therefore they can be neglected. Unfortunately our simulations have shown that the assumption of linkage equilibrium is seldom fulfilled for genetic algorithms. Therefore the variance decomposition method gives no good predictions for the heritability.

We will show in the next section that the regression technique can be used to control and guide the breeder genetic algorithm.

13 APPLICATIONS OF THE THEORY

From statistics and population genetics it is known that the regression coefficient should be a reliable estimate for heritability in the case of continuous fitness functions and large

populations. Therefore the first example is the minimization of the hyper-sphere. The BGA for continuous functions has been described in [Mühlenbein & Schlierkamp-Voosen, 1993b]. It uses a floating point representation. In figure 4 scatter diagrams of mid-parent and offspring at generation 1 and 30 are shown. In this example only discrete recombination is used, no mutation. It is easily seen that the whole population is moving towards the global minimum, which is 0 in this example. The regression coefficient is almost exactly one in both diagrams as predicted by the theory.

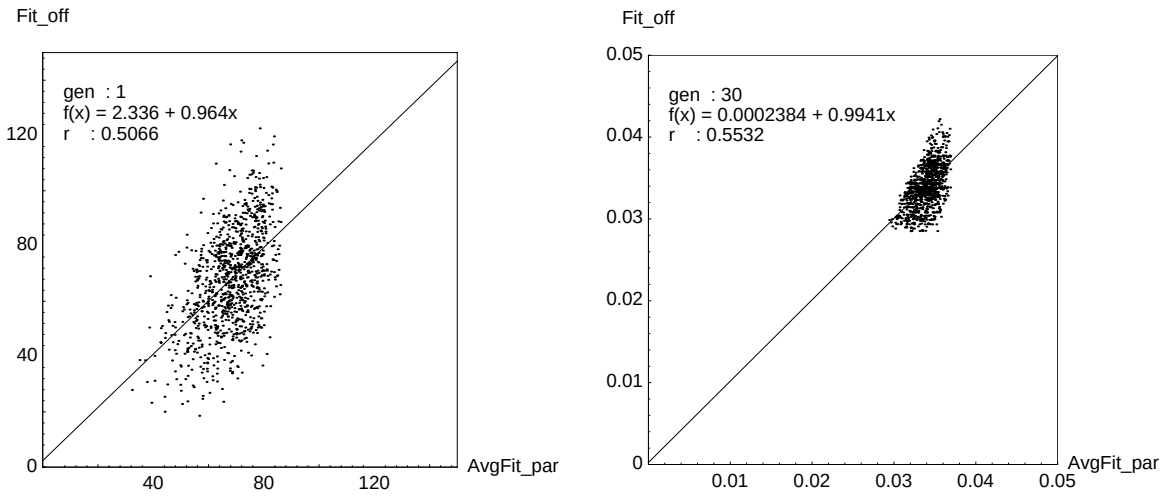

Figure 4: Scatter diagrams for generations 1 and 30 for the hypersphere. Only discrete recombination is used ($N = 1024, T = 0.5$).

In figure 5 the numerical values of the two different estimates for the heritability are shown ($R(t)/S(t)$ and the regression coefficient). Both estimates oscillate around 1 as predicted. The correlation coefficient is about 0.5. This is less than the maximum value possible, which is $\sqrt{0.5}$. The reason for this difference is the selection. The selection reduces the variance of the parents and therefore the correlation coefficient.

Results of a simulation run without selection confirm the theory. In this case the $R(t)/S(t)$ estimator cannot be used because $S(t)$ is 0. The regression coefficient can be computed as usual and remains 1. Furthermore the correlation coefficient is about $\sqrt{0.5}$ as predicted.

The above results are not restricted to simple unimodal functions. As the next example we take the highly multi-modal function which is known as Schwefel's function $F7$:

$$F_7 = \sum_{1}^{n} -x_i \; sin\left(\sqrt{|x_i|}\right) \qquad -500 \leq x_i \leq 500 \tag{64}$$

The theory predicts that the multi-modality of this function can be considered more or less as noise for the BGA. It should have no major influence on the regression coefficient. Indeed,

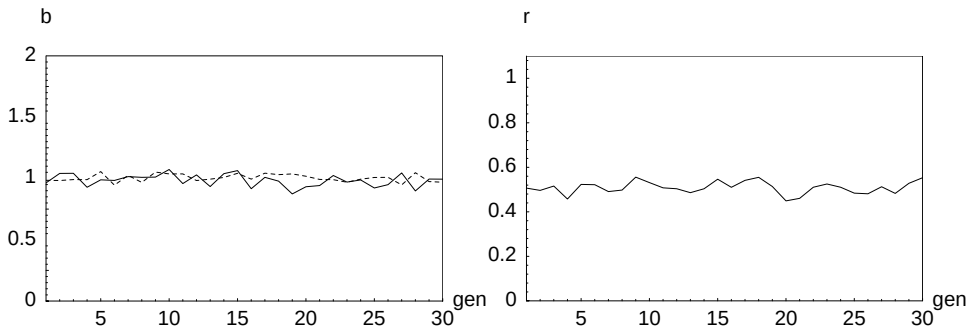

Figure 5: Heritability estimates (regression coefficient solid line, $R(t)/S(t)$ dashed line) and correlation coefficient r for the hyper-sphere ($N = 1024, T = 0.5$).

with random mating, the regression coefficient is 1 and the correlation coefficient between mid-parent and parent is about $\sqrt{0.5}$, just as for the hyper-sphere. Figure 6 shows a real BGA simulation run with selection, recombination *and* mutation. One clearly observes that the search is first driven by recombination, then by mutation. From generation 17 on, the regression coefficient substantially differs from the ratio estimator $R(t)/S(t)$. Now the search is mainly driven by the random operator mutation.

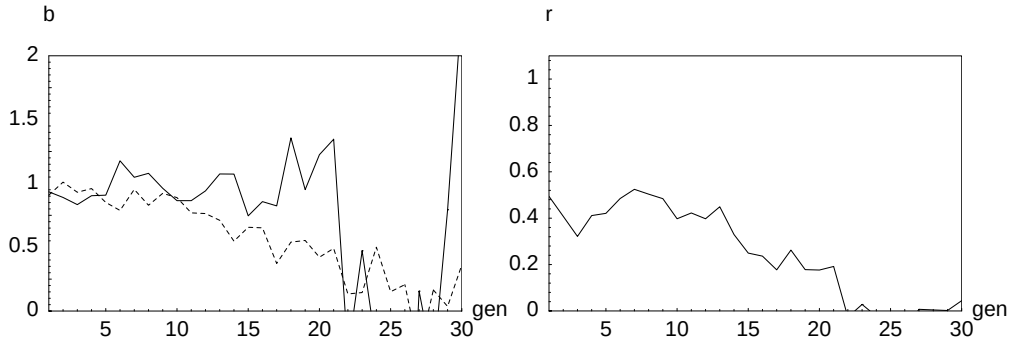

Figure 6: Heritability estimates b with mutation and recombination ($N = 256$). The correlation coefficient r drops to zero. The regression coefficient (solid line) and the ratio estimator (dashed line) are almost equal at the beginning. Then the ratio $R(t)/S(t)$ goes to zero whereas the regression coefficient remains high till generation 22.

Next we turn to binary functions. The following simple functions will be used

- $ONEMAX(n)$
- $PLATEAU(20, l)$
- $DECEP(10, 3)$

$PLATEAU(20, l)$ has a string length n of $20l$. An increase in fitness is allocated only if l consecutive bits starting at loci $1, l+1, 2l+1, ..$ are 1's. In each case, the fitness is increased by l. $DECEP(10, 3)$ is the deceptive function defined by Goldberg [1989].

In figure 7 the results of a BGA run are shown for $ONEMAX(64)$ with a truncation threshold of $T = 0.5$ and uniform crossover, but without mutation. The two heritability estimates coincide fairly well. They are about 1, as predicted. The correlation coefficient is about 0.5 till generation 14. This is less than the correlation coefficient without selection, which is $\sqrt{0.5}$. At the end of the run the correlation coefficient increases. This behavior indicates that the genotypes of the selected parents are becoming very similar. Therefore the offspring are very similar to both parents.

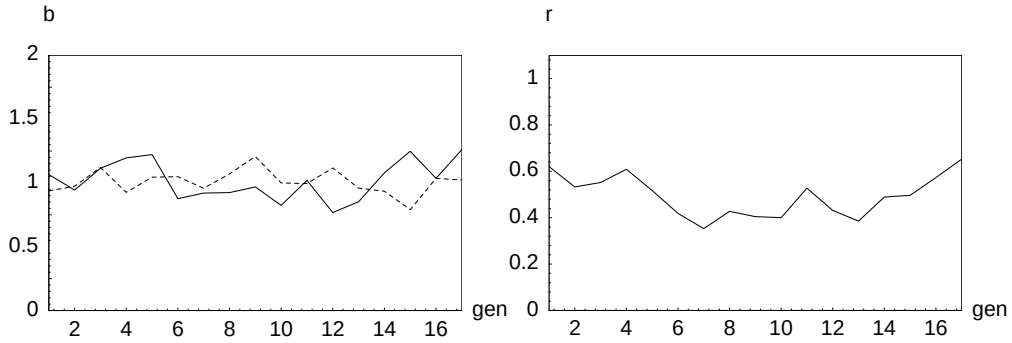

Figure 7: Heritability b estimates (regression coefficient solid line, $R(t)/S(t)$ dashed line) and correlation coefficient r with recombination only for ONEMAX(64) ($N = 128, T = 0.5$)

The next examples are $PLATEAU(20,3)$ and $PLATEAU(20,5)$. $PLATEAU(20,5)$ has a plateau of size 5, therefore it is more difficult to optimize. Without selection the regression coefficients for the two functions are about 0.7 and 0.4, the correlation coefficients are about 0.5 and 0.3. In figure 8 we have used a truncation threshold of $T = 0.5$. For both functions the regression coefficients are substantially higher than without selection. This indicates that selection is very effective for this fitness function. But note that the realized heritability $R(t)/S(t)$ is considerably smaller than the regression coefficient. For $PLATEAU(20,5)$ it substantially increases during the run.

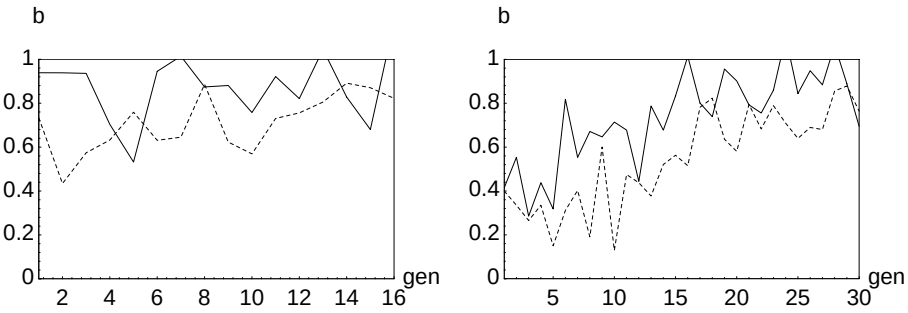

Figure 8: Heritability b estimates (regression coefficient solid line) for PLATEAU(20,3) and (20,5)

The last example is the deceptive function $DECEP(10,3)$. Without selection, the regression coefficient is about 0.5 and the correlation coefficient about 0.35. This is shown in figure 9.

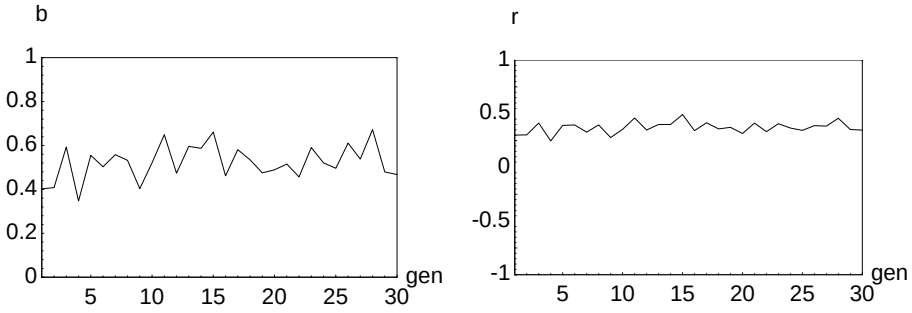

Figure 9: Heritability b and correlation r estimate with recombination for DECEP(10,3), no selection ($N = 256$)

The behavior radically changes with selection. If selection is applied, both the regression coefficient and the ratio estimator become erratic. Half of the time they are negative. This shows that selection and recombination work against each other. It confirms our earlier statement that genetic algorithms are not global optimizers.

To summarize this section:*For many continuous fitness functions the regression coefficient will be* 1*, the maximum possible. For binary functions the regression coefficient and the realized heritability give useful information about the complexity of the fitness landscape and how to guide the search of the breeder genetic algorithm.*

14 CONCLUSION

Evolutionary algorithms have been quite successfully applied to a number of difficult optimization problems. One of the major advantage of evolutionary algorithms is the fact that they can be intuitively understood. Unfortunately this advantage can turn to a disadvantage if researchers start to argue intuitively and not with mathematical rigour. Understanding why evolutionary algorithms work at all is almost as difficult as understanding natural evolution. The major difficulty lies in the fact that the algorithms combine two different search strategies, a random search by mutation and a biased search by recombination of the strings contained in the population.

We have presented in this chapter a theoretical analysis which is based on methods developed in population genetics. This analysis explains the behaviour of selection, mutation and recombination for problems which can be described by binary strings. The major part of the analysis deals with macroscopic variables. The genetic chance model is used only for estimating the heritability. The equation for the response to selection can be used for any representation. Only the results concerning the decomposition of the genetic variance are restricted to binary genes. The analysis has been used to design new genetic operators which do not have a counterpart in nature. Especially interesting is gene pool recombination. Its analysis shows that a genetic algorithm is not a global optimizer.

Applying genetic algorithms to general combinatorial optimization problems leads to the *genetic representation problem*. This means that a representation of the problem has to be found in order for mutation and recombination to create feasible offspring with high probability. This leads to unique and problem specific genetic operators mutation and recombination. We have presented a general design principle for such operators in this chapter. The principle states to maximize the product of heritability and variance.

Another solution to this problem may be a universal general genetic representation. This would make genetic algorithms still more similar to natural evolution. All natural systems use the same genetic mechanism, based on the interaction between DNA and RNA.

References

[AM94a] H. Asoh and H. Mühlenbein (1994a). Estimating the heritability by decomposing the genetic variance. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 866, pages 98–107. Springer-Verlag, New York.

[AM94b] H. Asoh and H. Mühlenbein (1994b). On the mean convergence time of evolutionary algorithms without selection and mutation. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 866, pages 88–97. Springer-Verlag, New York.

[Bäc93] Th. Bäck (1993). Optimal mutation rates in genetic search. In S. Forrest, editor, *5rd Int. Conf. on Genetic Algorithms*, pages 2–9, Morgan Kaufmann, San Mateo.

[BB91] R. K. Belew and L. Booker, (eds.) (1991) *Procedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo.

[BHS91] Th. Bäck, F. Hoffmeister, and H.-P. Schwefel (1991). A survey of evolution strategies. In R.K. Belew and L.B. Booker, editors, *Fourth Int. Conf. on Genetic Algorithms*, pages 2–9, Morgan Kaufmann, San Mateo.

[BlTh95] T. Blickle and L. Thiele (1995). A mathematical analysis of tournament selection. To appear in *Sixth Int. Conf. on Genetic Algorithms* Morgan Kaufmann, San Mateo.

[BJS66] H.J. Bremermann, J.Rogson, and S.Salaff (1966). Global properties of evolution processes. In H.H. Pattee, editor, *Natural Automata and Useful Simulations*, pages 3–42.

[BS93] Th. Bäck and H.-P. Schwefel (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1:1–24.

[Bul80] M. G. Bulmer (1980). *The Mathematical Theory of Quantitative Genetics*. Clarendon Press, Oxford.

[CK70] J. F. Crow and M. Kimura (1970). *An Introduction to Population Genetics Theory*. Harper and Row, New York.

[Cro86] J. F. Crow (1986). *Basic Concepts in Population, Quantitative and Evolutionary Genetics*. Freeman, New York.

[ERR94] A.E. Eiben, P.-E. Raue, and Zs. Ruttkay (1994). Genetic algorithms with multi-parent recombination. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 866, pages 78–87. Springer-Verlag, New York.

[EshSch93] Eshelman, L. J. and J. D. Schaffer (1993). Crossover's niche. In S.Forrest, ed., *Fifth Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, pp. 9–14.

[Esh91] L.J. Eshelman (1991). 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, Morgan-Kaufman, San Mateo.

[Fal81] D. S. Falconer (1981). *Introduction to Quantitative Genetics*. Longman, London.

[Fis58] R. A. Fisher (1958). *The Genetical Theory of Natural Selection*. Dover, New York.

[For93] S. Forrest, (ed.) (1993). *Procedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo.

[GD91] D.E. Goldberg and K. Deb (1991). A comparative analysis of selection schemes used in genetic algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93, Morgan-Kaufman, San Mateo.

[Gol89] D.E. Goldberg (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading.

[Gre86] J. J. Grefenstette (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems,Man and Cybernetics*, 16:122–128.

[Her92] Michael Herdy (1992). Reproductive Isolation as Strategy Parameter in Hierarchical Organized Evolution Strategies. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature*, pages 207–217, North-Holland, Amsterdam.

[Hol75] J.H. Holland (1975). *Adaptation in Natural and Artificial Systems*. Univ. of Michigan Press, Ann Arbor.

[Müh89] H. Mühlenbein (1989). Parallel genetic algorithm, population dynamics and combinatorial optimization. In H. Schaffer, editor, *3rd Int. Conf. on Genetic Algorithms*, pages 416–421, Morgan Kaufmann, San Mateo.

[Müh91] H. Mühlenbein (1991). Evolution in time and space - the parallel genetic algorithm. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337, Morgan-Kaufman, San Mateo.

[Müh92] H. Mühlenbein (1992). How Genetic Algorithms Really Work: Mutation and Hill-climbing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature*, pages 15–26, North-Holland, Amsterdam.

[MGSK88] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer (1988). Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–88.

[MSV93] H. Mühlenbein and D. Schlierkamp-Voosen (1993). Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization. *Evolutionary Computation*, 1:25–49.

[MSV94a] H. Mühlenbein and D. Schlierkamp-Voosen (1994a). The science of breeding and its application to the breeder genetic algorithm. *Evolutionary Computation*, 1:335–360.

[MSV94b] H. Mühlenbein and D. Schlierkamp-Voosen (1994b). The theory of breeding and the breeder genetic algorithm. In J. Stender, E. Hillebrand, and J. Kingdon, editors, *Genetic Algorithms in Optimisation, Simulation and Modelling*, Frontiers in Artificial Intelligence Applications, pages 27–64, IOS Press, Amsterdam.

[Nag82] H.N. Nagaraja (1982). Selection differentials. In E. Kotz, N.L. Johnson, and C.B. Read, editors, *Encyclopedia of Statistical Science*, pages 334–336, Wiley, New York.

[Nag92] T. Naglyaki (1992). *Introduction to Theoretical Population Genetics.* Springer, Berlin.

[Rao73] C.R. Rao (1973). *Linear Statisticcal Inference and Its Application.* Wiley, New York.

[Rec73] I. Rechenberg (1973). *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Information.* Fromman Verlag, Freiburg.

[Sch81] H.-P. Schwefel (1981). *Numerical Optimization of Computer Models.* Wiley, Chichester.

[Sch89] H. Schaffer, (ed.) (1989). *Procedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo.

[SE91] J.D. Schaffer and L.J. Eshelman (1991). On crossover as an evolutionary viable strategy. In R. K. Belew and L. Booker, editors, *Procedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68, Morgan Kaufmann, San Mateo.

[SVM94] D. Schlierkamp-Voosen and H. Mühlenbein (1994). Strategy adaptation by competing subpopulations. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science 866, pages 199–208. Springer-Verlag, New York.

[Sys89] G. Syswerda (1989). Uniform crossover in genetic algorithms. In H. Schaffer, editor, *3rd Int. Conf. on Genetic Algorithms*, pages 2–9, Morgan Kaufmann, San Mateo.

[Sys93] Syswerda, G. (1993). Simulated crossover in genetic algorithms, In L. D.Whitley, ed., *Foundations of Genetic Algorithms 2*, pp. 239–255, Morgan Kaufmann, San Mateo.

[VMC95] H.-M. Voigt, H. Mühlenbein and D. Cvetkovic (1995) Fuzzy recombination for the breeder genetic algorithm. To appear in *Sixth Int. Conf. on Genetic Algorithms* Morgan Kaufmann, San Mateo.