III

Evolutionary Computation and Beyond

RWCP Theoretical Foundation GMD Laboratory
HEINZ MÜHLENBEIN, THILO MAHNIG

Simulating evolution as seen in nature has been identified as one of the key computing paradigms for the new decade. Today evolutionary algorithms have been successfully used in a number of applications. These include discrete and continuous optimization problems, synthesis of neural networks, synthesis of computer programs from examples (also called genetic programming) and even evolvable hardware. But in all application areas problems have been encountered where evolutionary algorithms performed badly. Therefore a mathematical theory of evolutionary algorithms is urgently needed. Theoretical research has evolved from two opposed end; from the theoretical approach there are theories emerging that are getting closer to practice; from the applied side ad hoc theories have arisen that often lack theoretical justification.

In this chapter we concentrate on the analysis of evolutionary algorithms for optimization. The first section introduces the most popular algorithm, the *simple genetic algorithm.* This algorithm has many degrees of freedom, especially in the recombination scheme used. We show that all genetic algorithms behave very similar, if recombination is done without selection a sufficient number of times before the next selection step. This conceptual algorithm we approximate by the *Univariate Marginal Distribution Algorithm* $UMDA$, which is analyzed in Section 2. We compute the difference equation for the univariate marginal distributions under the assumption of proportionate selection. This equation has

Foundations of Real-World Intelligence.
Yoshinori Uesaka et al. (eds.).

been proposed in populations genetics by Sewall Wright as early as 1937 (1970). This is an independent confirmation of our claim that $UMDA$ approximates any genetic algorithm. Using *Wright's equation* we show that $UMDA$ solves a *continuous optimization problem.* The function to be optimized is given by the average fitness of the population.

Proportionate selection is far too weak for optimization. This has been recognized very early in breeding of livestock. *Artificial selection* as done by breeders is a much better model for optimization than *natural selection* modelled by proportionate selection. Unfortunately an exact mathematical analysis of efficient artificial selection schemes seems impossible. Therefore breeders have developed an approximate theory, using the concepts of regression of offspring to parent, heritability and response to selection. This theory is discussed in Section 3. At the end of the section numerical results are shown which show the strength and the weakness of $UMDA$ as a numerical optimization method.

$UMDA$ optimizes very efficient some difficult optimization problems, but it fails on some simple problems. For these problems higher order marginal distributions are necessary which capture the nonlinear dependency between variables. In Section 4.1 $UMDA$ is extended to the *Factorized Distribution Algorithm FDA.* We prove convergence of the algorithm to the global optima if *Boltzmann selection* is used. The theory of factorization connects FDA with the theory of *graphical models* and *Bayesian networks.* We derive a new adaptive Boltzmann selection schedule SDS using ideas from the science of breeding.

In Section 5.1 we use results from the theory of Bayesian networks for the *Learning Factorized Distribution Algorithm $LFDA$*, which learns a factorization from the data. We make a preliminary comparison between the efficiency of FDA and $LFDA$.

In Section 6 we describe the *system dynamics approach to optimization.* The difference equations obtained for $UMDA$ are iterated until convergence. Thus the continuous optimization problem is mathematically solved without using a population of points at all. We present numerical results for three different system dynamics equations. They consists of Wright's equation, the *diversified replicator equation* and a modified version of Wright's equation which converges faster.

In the final section we classify the different evolutionary computation methods presented. The classification criterion is whether a microscopic or a macroscopic model is used for selection and/or recombination.

1 Analysis of the Simple Genetic Algorithm

In this section we investigate the standard genetic algorithm, also called the Simple Genetic Algorithm (SGA). The algorithm is described by Holland (1975/1992) and Goldberg (1989). It consists of

- fitness proportionate selection
- recombination/crossover
- mutation

Here we will analyze selection and recombination only. Mutation is considered to be a background operator. It can be analyzed by known techniques from stochastics (Mühlenbein & Schlierkamp-Voosen, 1994; Mühlenbein, 1997).

There have been many claims concerning the optimization power of *SGA*. Most of them are based on a rather qualitative application of the *schema theorem.* We will show the shortcomings of this approach. Our analysis is based on techniques used in population genetics. The analysis reveals that an exact mathematical analysis of *SGA* is possible for small problems only. For a binary problem of size n the exact analysis needs the computation of 2^n equations. But we propose an approximation often used in population genetics. The approximation assumes that the gene frequencies are in *linkage equilibrium.* The main result is that *any genetic algorithm can be approximated by an algorithm using n parameters only, the univariate marginal gene frequencies.*

1.1 Definitions

Let $\mathbf{x} = (x_1, \ldots, x_n)$ denote a binary vector. For notational simplicity we restrict the discussion to binary variables $x_i \in \{0, 1\}$. We use the following conventions. Capital letters X_i denote variables, small letters x_i assignments.

Definition 1.1. *Let a function $f : \boldsymbol{X} \to R^{\geq 0}$ be given. We consider the optimization problem*

$$\mathbf{x}_{opt} = \operatorname{argmax} f(\mathbf{x}) \tag{1.1}$$

We will use $f(\mathbf{x})$ as the fitness function for the *SGA*. We will investigate two widely used recombination/crossover schemes.

Definition 1.2. *Let two strings* $\mathbf{x}$ *and* $\mathbf{y}$ *be given. In* one-point crossover *the string* $\mathbf{z}$ *is created by randomly choosing a crossover point* $0 < l < n$ *and setting* $z_i = x_i$ *for* $i \leq l$ *and* $z_i = y_i$ *for* $i > l$. *In* uniform crossover z_i *is randomly chosen with equal probability from* $\{x_i, y_i\}$.

Definition 1.3. *Let $p(\mathbf{x},t)$ denote the probability of $\mathbf{x}$ in the population at generation t. Then $p_i(x_i,t) = \sum_{\mathbf{x},X_i=x_i} p(\mathbf{x},t)$ defines a univariate marginal distribution.*

We often write $p_i(x_i)$ if just one generation is discussed. In this notation the average fitness of the population and the variance is given by

$$\begin{aligned} \bar{f}(t) &= \sum_x p(\mathbf{x},t) f(\mathbf{x}) \\ V(t) &= \sum_x p(\mathbf{x},t) \left(f(\mathbf{x}) - \bar{f}(t)\right)^2 \end{aligned}$$

The *response to selection* $R(t)$ is defined by

$$R(t) = \bar{f}(t+1) - \bar{f}(t) \tag{1.2}$$

1.2 Proportionate Selection

Proportionate selection changes the probabilities according to

$$p(\mathbf{x},t+1) = p(\mathbf{x},t)\frac{f(x)}{\bar{f}(t)} \tag{1.3}$$

Lemma 1.1. *For proportionate selection the response is given by*

$$R(t) = \frac{V(t)}{\bar{f}(t)} \tag{1.4}$$

Proof: We have

$$R(t) = \sum_x p(x,t)\frac{f^2(x)}{\bar{f}(t)} - \bar{f}(t) = \frac{V(t)}{\bar{f}(t)} \tag{1.5}$$

□

With proportionate selection the average fitness never decreases. This is true for every rational selection scheme.

1.3 Recombination

For the analysis of recombination we introduce a special distribution.

Definition 1.4. *Robbins' proportions are given by the distribution π*

$$\pi(x,t) := \prod_{i=1}^{n} p_i(x_i,t) \tag{1.6}$$

A population in Robbins' proportions is also called to be in linkage equilibrium.

Geiringer (1944) has shown that all reasonable recombination schemes lead to the same limit distribution.

Theorem 1.1 (Geiringer). *Recombination does not change the univariate marginal frequencies, i.e.* $p_i(x_i, t+1) = p_i(x_i, t)$*. The limit distribution of any complete recombination scheme is Robbins' proportions* $\pi(\mathbf{x})$*.*

Complete recombination means that for each subset S of $\{1, \dots, n\}$, the probability of an exchange of genes by recombination is greater than zero. Convergence to the limit distribution is very fast. We will prove this for $n = 2$ loci.

Theorem 1.2. *Let* $D(t) = p(0,0,t)p(1,1,t) - p(0,1,t)p(1,0,t)$*. If there is no selection then we have for two loci and uniform crossover*

$$D(t) = (-1)^{|x|^2}\big(p(\mathbf{x}, t) - p_1(x_1, 0)p_2(x_2, 0)\big). \tag{1.7}$$

$|x|^2$ *denotes the number of ones in* $\mathbf{x}$*.* $p_i(x_i, 0)$ *denotes the univariate marginal frequency at* $t = 0$*. The factor* $D(t)$ *is halved each generation*

$$D(t+1) = \frac{1}{2}D(t). \tag{1.8}$$

Proof: Without selection the univariate marginal frequencies are independent of t, because in an infinite population a recombination operator does not change them. Then from

$$\begin{aligned}
&p(1,1,t) - p_1(1,0)p_2(1,0)\\
&\qquad = p(1,1,t) - \big(p(1,0,t) + p(1,1,t)\big)\big(p(0,1,t) + p(1,1,t)\big)\\
&\qquad = p(1,1,t) - p(0,1,t)p(1,0,t) - p(1,1,t)(1 - p(0,0,t)) = D(t)
\end{aligned}$$

we obtain Equation 1.7 for $\mathbf{x} = (1,1)$. The other cases are proven in the same way.

The gene frequencies after recombination are obtained as follows. We only consider $p(1,1,t)$. The probability of $p(1,1,t+1)$ can be computed from the probability that recombination generates string $(1,1)$. The probability is given by

$$\begin{aligned}
p(1,1,t+1) &= p(1,1,t) \cdot \big(\tfrac{1}{2}p(0,0,t) + p(0,1,t) + p(1,0,t) + p(1,1,t)\big)\\
&\quad + \frac{1}{2}p(0,1,t)p(1,0,t)\\
&= p(1,1,t) - \tfrac{1}{2}\big(p(1,1,t)p(0,0,t) - p(0,1,t)p(1,0,t)\big)\\
&= p(1,1,t) + (-1)^{|\mathbf{x}|^2+1}\frac{1}{2}D(t).
\end{aligned}$$

By computing D(t+1) Equation 1.8 is obtained. □

We will use as a measure for the deviation from Robbins' proportions the mean square error $DSQ(t)$

$$DSQ(t) = \sum_{\mathbf{x}} \big(p(\mathbf{x},t) - p_1(x_1)p_2(x_2)\big)^2. \tag{1.9}$$

From the above theorem we obtain

Corollary 1.1. *For two loci the mean square error is reduced each step by one fourth*

$$DSQ(t+1) = \frac{1}{4} DSQ(t)$$

For more than 2 loci the equations for uniform crossover and one-point crossover get more complicated. Uniform crossover converges faster to linkage equilibrium, because it mixes the genes much more than one-point crossover.

In a finite population linkage equilibrium cannot be exactly achieved. Let us take the uniform distribution as example. Here linkage equilibrium is given by $p(\mathbf{x}) = 2^{-n}$. This value can only be obtained if the size of the population is substantial larger than 2^n. In a finite population we observe first a fast decrease of linkage disequlibrium. Then DSQ slowly increases due to stochastic fluctuations by *genetic drift.* Ultimately the population will consist of one genotype only. Genetic drift has been analyzed by Asoh and & Mühlenbein (1994b). It will not be considered here.

1.4 Selection and Recombination

We have shown that the average $\bar{f}(t)$ never decreases after selection and that any complete recombination scheme moves the genetic population to Robbins' proportions. Now the question arises: What happens if recombination is applied *after* selection. The answer is very difficult. The problem still puzzles populations genetics (Nagylaki, 1992).

Formally the difference equations can be easily written. Let a recombination distribution R be given. $R_{x,yz}$ denotes the probability that y and z produce x after recombination. Then

$$p(\mathbf{x}, t+1) = \sum_{y,z} R_{x,yz} p^s(\mathbf{y}) p^s(\mathbf{z}) \tag{1.10}$$

$p^s(x)$ denotes the probability of string x after selection. For n loci the recombination distribution R consists of $2^n * 2^n$ parameters. Recently Christiansen and Feldman (1998) have written a survey about the mathematics of selection and recombination from the viewpoint of population genetics. A new techniques to obtain the equations has been developed by Vose (1999). In both frameworks one needs a computer program to compute the equations for a given fitness function.

We discuss the problem for a special case only, uniform crossover for $n = 2$ loci.

Theorem 1.3. *For proportionate selection and uniform crossover the gene frequencies obey the following difference equation*

$$p(\mathbf{x}, t+1) = \frac{f(\mathbf{x})}{\bar{f}(t)} p(\mathbf{x}, t) + (-1)^{|\mathbf{x}|^2+1} \frac{1}{2} \frac{D_s(t)}{\bar{f}(t)^2}. \tag{1.11}$$

$|\mathbf{x}|^2$ *denotes the number of ones in* $\mathbf{x}$. $\bar{f}(t) = \sum_{\mathbf{x}} p(\mathbf{x}, t) f(\mathbf{x})$ *is the average fitness of the population; and* $D_s(t)$ *is defined as*

$$\begin{aligned} D_s(t) = {} & f(0,0) f(1,1) p(0,0,t) p(1,1,t) \\ & - f(0,1) f(1,0) p(1,0,t) p(0,1,t) \end{aligned} \tag{1.12}$$

Proof: For proportionate selection the gene frequencies $p^s(\mathbf{x}, t)$ after selection are given by

$$p^s(\mathbf{x}, t) = \frac{f(\mathbf{x})}{\bar{f}(t)} p(\mathbf{x}, t).$$

Now we pair randomly between the selected parents and count how often genotype $\mathbf{x}$ arises after uniform crossover. Taking $\mathbf{x} = (0,0)$ as an example, and computing the probabilities of mating, we obtain

$$\begin{aligned} & p(0{,}0, t+1) = \\ & \quad p^s(0,0,t) \left(p^s(0,0,t) + p^s(0,1,t) + p^s(1,0,t) + \frac{1}{2} p^s(1,1,t) \right) \\ & \quad + \frac{1}{2} p^s(0,1,t) p^s(1,0,t) \end{aligned}$$

Using the fact that $p^s(0,0,t) + p^s(0,1,t) + p^s(1,0,t) + p^s(1,1,t) = 1$ we obtain the theorem for $\mathbf{x} = (0,0)$. The remaining equations are obtained in the same manner. □

A mathematical analysis of the mathematical properties of n loci systems is difficult. For a problem of size n we have 2^n equations. Furthermore the equations depend on the recombination operator used! If the gene frequencies remain in linkage equilibrium, then only n equations are needed for the marginal frequencies. Thus the crucial question is: Does the optimization process gets worse because of this simplification? The answer is no. We provide evidence for this statement by citing a theorem from (Mühlenbein, 1997). It shows that the univariate marginal frequencies are the same for all recombination schemes if applied to the same distribution $p(\mathbf{x}, t)$.

Theorem 1.4. *For* any *complete recombination/crossover scheme used after proportionate selection the univariate marginal frequencies are de-*

termined by

$$p(x_i, t+1) = \sum_{\mathbf{x}|X_i=x_i} \frac{p(\mathbf{x},t)f(\mathbf{x})}{\bar{f}(t)}. \tag{1.13}$$

Proof: After selection the univariate marginal frequencies are given by

$$p^s(x_i,t) = \sum_{\mathbf{x}|X_i=x_i} p^s(\mathbf{x},t) = \sum_{\mathbf{x}|X_i=x_i} \frac{p(\mathbf{x},t)f(\mathbf{x})}{\bar{f}(t)}.$$

Now the selected individuals are randomly paired. Since complete recombination does not change the allele frequencies, these operators do not change the univariate marginal frequencies. Therefore

$$p_i(x_i, t+1) = p_i^s(x_i, t).$$

□

1.5 Schema Analysis Demystified

Many of the more intuitive arguments about the behavior of genetic algorithm are based on the analysis of "schemata" and their evolution in a population. The theory has been developed by Holland (1975/1992). By using probability distributions and an ideal schema equation we demonstrate by a simple example that the more intuitive conclusions about the proliferation of schemata can be misleading. Our analysis is based on an exact solution of the probability distribution for proportionate selection.

Definition 1.5. *Let $p(\mathbf{x},t)$ denote the probability of $\mathbf{x}$ in the population at generation t. Let $\mathbf{x}_s = (x_{s_1}, \ldots, x_{s_i}) \subset \{x_1, \ldots, x_n\}$. Thus $\mathbf{x}_s$ denotes a subvector of $\mathbf{x}$ defined by the indices $s_1, \ldots, s_i$. Then the probability of schema $H(s)$ is defined by*

$$p(H(s),t) = \sum_{X|X_s=x_s} p(\mathbf{x},t) \tag{1.14}$$

The summation is done by fixing the values of $\mathbf{x}_s$. Thus the probability of a schema is just the corresponding marginal distribution $p(\mathbf{x}_s)$. If $\mathbf{x}_s$ consists of a single element only, we have a univariate marginal distribution.

SGA uses fitness proportionate selection, i.e. the probability of $\mathbf{x}$ being selected is given by

$$p^s(\mathbf{x},t) = p(\mathbf{x},t)\frac{f(\mathbf{x})}{\bar{f}(t)} \tag{1.15}$$

$\bar{f}(t) = \sum_x p(\mathbf{x},t)f(\mathbf{x})$ is the average fitness of the population. Let us now assume that we have an algorithm which generates new points according to the distribution of selected points, i.e.

$$p(\mathbf{x}, t+1) = p(\mathbf{x}, t)\frac{f(\mathbf{x})}{\bar{f}(t)} \tag{1.16}$$

$p(\mathbf{x}, t+1)$ can be seen as the ideal probability distribution of *SGA*.

Definition 1.6. *The fitness of schema $H(\boldsymbol{s}_i)$ is defined by*

$$f(H(\boldsymbol{s}_i), t) = \sum_{X|X_s = x_s} \frac{p(\mathbf{x}, t)}{p(H(\boldsymbol{s}), t)} f(\mathbf{x}) \tag{1.17}$$

Theorem 1.5 (Schema Theorem). *The probability of schema $H(\boldsymbol{s})$ is given by*

$$p(H(\boldsymbol{s}), t+1) = p(H(\boldsymbol{s}, t))\frac{f(H(\boldsymbol{s}), t)}{\bar{f}(t)} \tag{1.18}$$

Holland ((1975/1992) Theorem 6.2.3) computed for SGA (a genetic algorithm with recombination and mutation) an inequality

$$p(H(\boldsymbol{s}), t+1) \geq (1-\delta) p(H(\boldsymbol{s}, t))\frac{f(H(\boldsymbol{s}), t)}{\bar{f}(t)} \tag{1.19}$$

δ is a small factor. The inequality complicates the analysis. Equation (1.17 is obviously the *ideal starting equation* for Holland's schema analysis.

The mathematical difficulty of using the inequality (1.19) to estimate the distribution of schemata lies in the fact that the fitness of a schema depends on $p(\mathbf{x}, t)$, i.e the distribution of the genotypes of the population. This is a defining fact of Darwinian *natural selection*. The fitness is always relative to the current population. To cite a proverb: *the one-eyed is the king of the blinds.*

Thus an application of the inequality (1.19) is not possible without computing $p(\mathbf{x}, t)$. Goldberg (1989) circumvented this problem by assuming

$$p(H(\boldsymbol{s}), t) \geq (1+c)\bar{f}(t) \tag{1.20}$$

With this assumption we estimate $p(H(\boldsymbol{s}), t) \geq (1+c)^t p(H(\boldsymbol{s}), 0)$. But the assumption can never be fulfilled for all t. When approaching an optimum, the fitness of all schemata in the population will be only $1 \pm \epsilon$ away from the average fitness. Here proportionate selection gets difficulties.

The typical folklore which arose from the schema analysis is nicely summarized by Ballard ((1997), p.270). He is not biased towards or against genetic algorithms. He just cites the commonly used arguments:

- *Short schemata have a high probability of surviving the genetic operations.*
- *Focusing on short schemata that compete shows that, over the short run, the fittest are increasing at an exponential rate.*
- *Ergo, if all of the assumptions hold (we cannot tell whether they do, but we suspect they do), GAs are optimal.*

We will not investigate the optimality argument, because we will show that the basic conclusion of exponential increasing schemata does not hold.

It turns out that equation 1.16 for proportionate selection admits an analytical solution.

Theorem 1.6 (Convergence). *The distribution* $p(\mathbf{x},t)$ *for proportionate selection is given by*

$$p(\mathbf{x},t) = \frac{p(\mathbf{x},0)f(\mathbf{x})^t}{\sum_y p(\mathbf{y},0)f(\mathbf{y})^t} \tag{1.21}$$

Let $\mathcal{M}$ *be the set of global optima, then*

$$\lim_{t\to\infty} p(\mathbf{x},t) = \begin{cases} 1/|\mathcal{M}| & \mathbf{x} \in \mathcal{M} \\ 0 & \textit{else} \end{cases} \tag{1.22}$$

Proof: The proof is by induction. The assumption is fulfilled for $t = 1$. Then

$$\begin{aligned} p(\mathbf{x},t+1) &= \frac{p(\mathbf{x},0)f(\mathbf{x})^{t+1}}{\sum_y p(\mathbf{y},0)f(\mathbf{y})^{t+1}} \\ &= \frac{p(\mathbf{x},0)f(\mathbf{x})^t}{\bar{f}(t)} \frac{f(\mathbf{x})}{\sum_y \frac{p(\mathbf{y},0)f(\mathbf{y})^t \cdot f(\mathbf{y})}{\bar{f}(t)}} \\ &= \frac{p(\mathbf{x},0)f(\mathbf{x})^{t+1}}{\sum_y p(\mathbf{y},0)f(\mathbf{y})^{t+1}} \end{aligned}$$

Let $\mathbf{x}_{max} \in \mathcal{M}$ and $f(\mathbf{x}) < f(\mathbf{x}_{max})$. Then

$$\frac{p(\mathbf{x},t)}{p(\mathbf{x}_{max},t)} = \frac{p(\mathbf{x},0)f(\mathbf{x})^t}{p(\mathbf{x}_{max},0)f(\mathbf{x}_{max})^t} \to 0$$

□

This shows that our algorithm is ideal in the sense that it even converges to the set of global optima. Equation 1.21 was already used by Goldberg and Deb ((1991)).

By using equation (1.21) we can make a correct schema analysis. We compute the probabilities of all schemata. We just discuss the interesting case of a *deceptive function*. We take the 3-bit deceptive function defined

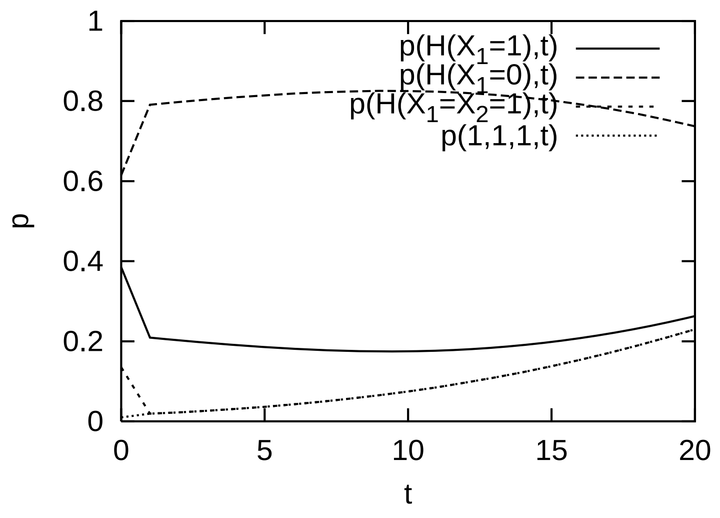

FIGURE 1 Evolution of some schemata

by

$$
\begin{aligned}
decep(\mathbf{x}) = 0.9 - 0.1(x_1 + x_2 + x_3) \\
- 0.7(x_1x_2 + x_2x_3 + x_1x_3) + 2.5x_1x_2x_3
\end{aligned}
$$

The function is called deceptive because the global optimum $(1,1,1)$ is isolated, whereas the local optimum $(0,0,0)$ is surrounded by strings of high fitness. We now look at the behavior of some schemata.

Definition 1.7. *A schema is called optimal if its defining string s is contained in an optimal string.*

In our example $H(X_1 = 1)$ and $H(X_1 = X_2 = 1)$ are optimal schemata. They are displayed in Figure 1. We see that the probability of the optimal schema $p(H(X_1 = 1)$ decreases for about 8 generations, then it increases fairly slowly. This behavior is contrary to the simple interpretation of the evolution of schemata. Schema $H(X_1 = X_2 = 1)$ decreases even dramatically at the first generation. Then its probability is almost identical to the probability of the optimum $(1,1,1)$.

We summarize the results. All complete recombination schemes lead to the same univariate marginal distributions after one step of selection and recombination. If recombination is used for a number of times without selection, then the genotype frequencies converge to linkage equilibrium. This means that *all genetic algorithms are identical if after after one selection step recombination is done without selection a sufficient number* *of times.* This fundamental algorithm keeps the population in linkage equilibrium. In the next section we will analyze this algorithm.

2 The Univariate Marginal Distribution Algorithm *UMDA*

The univariate marginal distribution algorithm *UMDA* generates new points according to $p(\mathbf{x}, t) = \prod_{i=1}^{n} p_i^s(x_i, t)$. Thus *UMDA* keeps the gene frequencies in linkage equilibrium. This makes a mathematical analysis possible. We derive a difference equation for proportionate selection. This equation has already been proposed by Sewall Wright in 1937 (1970). Wright's equation shows that *UMDA* is trying to solve a continuous optimization problem. The continuous function to be optimized is the average fitness of the population $W(\mathbf{p})$. The variables are the univariate marginal distributions. In a fundamental theorem we show the relation between the attractors of the continuous problem and the local optima of the fitness function $f(\mathbf{x})$.

2.1 Definition of *UMDA*

Instead of performing recombination a number of times in order to converge to linkage equilibrium, one can achieve this in one step by *gene pool recombination* (Mühlenbein & Voigt, 1996). In gene pool recombination a new string is computed by randomly taking for each loci a gene from the distribution of the selected parents. This means that gene x_i occurs with probability $p_i^s(x_i)$ in the next population. $p_i^s(x_i)$ is the distribution of x_i in the selected parents. New strings $\mathbf{x}$ are generated according to the distribution

$$p(\mathbf{x}, t+1) = \prod_{i=1}^{n} p_i^s(x_i, t) \tag{2.1}$$

One can simplify the algorithm still more by directly computing the univariate marginal frequencies from the data. Then Equation 2.1 can be used to generate new strings. This method is used by *UMDA*.

UMDA

- **STEP 0:** Set $t \Leftarrow 1$. Generate $N \gg 0$ points randomly.
- **STEP 1:** Select $M \leq N$ points according to a selection method. Compute the marginal frequencies $p_i^s(x_i, t)$ of the selected set.
- **STEP 2:** Generate N new points according to the distribution $p(\mathbf{x}, t+1) = \prod_{i=1}^{n} p_i^s(x_i, t)$. Set $t \Leftarrow t + 1$.

- **STEP 3:** If termination criteria are not met, go to STEP 1.

For proportionate selection we need the average fitness of the population $\bar{f}(t)$. We consider $\bar{f}(t)$ as a function which depends on $p(x_i)$. To emphasize this dependency we write

$$W(p_1(X_1=0), p(X_1=1), \dots, p_n(X_n=1)) := \bar{f}(t) \tag{2.2}$$

W formally depends on $2n$ parameters. $p_i(X_i = 1)$ and $p_i(X_i = 0)$ are considered as two independent parameters despite the constraint $p_i(X_i = 0) = 1 - p_i(X_i = 1)$. We abbreviate $p_i := p_i(X_i = 1)$. If we insert $1 - p_i$ for $p_i(X_i = 0)$ into W, we obtain $\tilde{W}$. $\tilde{W}$ depends on n parameters. Now we can formulate the main theorem.

Theorem 2.1. *For infinite populations and proportionate selection the difference equations for the gene frequencies used by UMDA are given by*

$$p_i(x_i, t+1) = p_i(x_i, t)\frac{\bar{f}_i(x_i, t)}{W(t)} = p_i(x_i, t)\frac{\frac{\partial W}{\partial p_i(x_i)}}{W(t)} \tag{2.3}$$

where $\bar{f}_i(x_i, t) = \sum_{\mathbf{x}, X_i = x_i} f(\mathbf{x}) \prod_{j \neq i}^{n} p(x_j, t)$. *The equation can also be written as*

$$p_i(t+1) = p_i(t) + p_i(t)(1 - p_i(t))\frac{\frac{\partial \tilde{W}}{\partial p_i}}{\tilde{W}(t)} \tag{2.4}$$

The response $R(t)$ *is given by*

$$\begin{aligned} R(t) = {} & \frac{V_A(t)}{\tilde{W}} + \frac{1}{2}\sum_{i \neq j} \frac{\alpha_i * \alpha_j}{\tilde{W}^2} \frac{\partial^2 W}{\partial p_i \partial p_j} \\ & + \frac{1}{3!} \sum_{i \neq j, j \neq k, i \neq k} \frac{\alpha_i * \alpha_j * \alpha_k}{\tilde{W}^3} \frac{\partial^3 \tilde{W}}{\partial p_i \partial p_j \partial p_k} + \dots. \end{aligned} \tag{2.5}$$

$$VA(t) = \sum_i p_i(1, t)(f_i(1, t) - \tilde{W})^2 + p_i(0, t)(f_i(0, t) - \tilde{W})^2 \tag{2.6}$$

$$\alpha_i = p_i(t)(1 - p_i(t))\frac{\partial \tilde{W}}{\partial p_i}$$

$VA(t)$ is called the **additive genetic variance**. *Furthermore the average fitness never decreases*

$$W(t+1) \geq W(t) \tag{2.7}$$

Proof: Equation 2.3 has been proven in (Mühlenbein, 1997). We have to prove Equation 2.4. Note that

$$p_i(t+1) - p_i(t) = p_i(t)\frac{\bar{f}_i(x_i = 1, t) - \tilde{W}(t)}{\tilde{W}(t)}$$

Obviously we have

$$\frac{\partial \tilde{W}}{\partial p_i} = \bar{f}(x_i = 1, t) - \bar{f}(x_i = 0, t)$$

From $p_i(t)\bar{f}_i(x_i = 1, t) + (1 - p_i(t))\bar{f}_i(x_i = 0, t) = \tilde{W}(t)$, we obtain

$$\bar{f}(x_i = 1, t) - \tilde{W}(t) - (1 - p_i(t))\bar{f}(x_i = 1, t) + (1 - p_i(t))\bar{f}(x_i = 0, t) = 0$$

This gives

$$\bar{f}_i(x_i = 1, t) - \tilde{W}(t) = (1 - p_i(t))\frac{\partial \tilde{W}}{\partial p_i}$$

Inserting this equation into the difference equation gives Equation 2.4. Equation 2.5 is just the multi-dimensional Taylor expansion. The first term follows from

$$\begin{aligned}
\sum_i (p_i(t+1) - p_i(t))\frac{\partial \tilde{W}}{\partial p_i} &= \sum_i p_i(t)(1 - p_i(t))\left(\frac{\partial \tilde{W}}{\partial p_i}\right)^2 \\
&= \sum_i p_i(t)(f_i(1,t) - \tilde{W})(f_i(1,t) - \tilde{W} + \tilde{W} - f_i(0,t)) \\
&= \sum_i p_i(t)(f_i(1,t) - \tilde{W})^2 + (1 - p_i(t))(f_i(0,t) - \tilde{W}) = VA(t)
\end{aligned}$$

$\square$

The above equations completely describe the dynamics of UMDA with proportionate selection. Mathematically UMDA performs gradient ascent in the landscape defined by W or $\tilde{W}$.

Equation 2.4 is especially suited for the theoretical analysis. It is called *Wright's equation* because it has been proposed by Wright in 1937. Wright's (1970) remarks are still valid today:

> The appearance of this formula is deceptively simple. Its use in conjunction with other components is not such a gross oversimplification in principle as has sometimes been alleged ... Obviously calculations can be made only from rather simple models, involving only a few loci or simple patterns of interaction among many similarly behaving loci... Apart from application to simple systems, the greatest significance of the general formula is that its form brings out properties of systems that would not be apparent otherwise.

The restricted application lies in the following fact. In general the difference equations need the evaluation of 2^n terms. The computational complexity can be drastically reduced if the fitness function has a special form.

Example 2.1. $f(x) = \sum_i a_i x_i, \quad x_i \in \{0,1\}$
After some tedious manipulations one obtains:

$$\begin{aligned} W(\mathbf{p}) &= \sum_i a_i p_i(1) \\ \frac{\partial W}{\partial p_i(1)} &= a_i + \sum_{j \neq i} a_j p_j(1) \end{aligned}$$

This gives the difference equation

$$\Delta p_i(1) = p_i(1,t)(1 - p_i(1,t)) \frac{a_i}{\sum_i a_i p_i(1,t)} \tag{2.8}$$

Noting that $\frac{\partial \tilde{W}}{\partial p_i(1)} = a_i$ we have proving nothing else than Wright's equation. This equation has been approximately solved in (Mühlenbein & Mahnig, 1999a).

This example shows that the expressions for W and its derivatives can be surprisingly simple. $W(\mathbf{p})$ can be obtained from $f(\mathbf{x})$ by exchanging x_i with $p_i(1)$. But the formal derivation of $W(\mathbf{p})$ cannot be obtained from the simplified $W(\mathbf{p})$ expression.

Another interesting example is a multiplicative function.

Theorem 2.2. *For a multiplicative function $f(\mathbf{x}) = \prod_{i=1}^n f_i(x_i)$ we have*

$$R(t) = \frac{V(t)}{\tilde{W}} = S(t) \tag{2.9}$$

Proof: The proof is technically somewhat complicated. Therefore we just sketch the proof for $n = 2$. We set $p_1 = p_1(x_1,t)$ and $p_2 = p_2(x_2,t)$. Using equation 2.3 we obtain

$$\begin{aligned} p(\mathbf{x}, t+1) &= \frac{p_1 \bar{f}_1(x_1,t) p_2 \bar{f}(x_2,t)}{W^2} \\ \bar{f}_1(x_1,t) &= p_2 f(x_1,x_2) + (1-p_2) f(x_1, 1-x_2) \\ \bar{f}_2(x_2,t) &= p_1 f(x_1,x_2) + (1-p_1)) f(1-x_1, x_2) \end{aligned}$$

After some manipulations we obtain

$$\bar{f}_1(x_1,t)\bar{f}_2(x_2,t) = f(x_1,x_2)W$$

and finally

$$p(\mathbf{x}, t+1) = p(\mathbf{x}, t)\frac{f(\mathbf{x})}{W}$$

From Lemma 1.1 we obtain $R(t) = V(t)/W$.

$\square$

We will investigate the computation of W and its gradient in the following section.

2.2 Computing the Average Fitness

Wright is also the originator of the landscape metaphor now popular in evolutionary computation and population genetics. Unfortunately Wright used two quite different definitions for the landscape, apparently without realizing the fundamental distinction between them. The first landscape describes the relation between the genotypes and their fitness, while the second describes the relation between the allele frequencies in a population and its mean fitness.

The first definition is just the fitness function $f(\mathbf{x})$ used in evolutionary computation, the second one is the average fitness $\tilde{W}(\mathbf{p})$. The second definition is much more useful, because it lends to a quantitative description of the evolutionary process, i.e. Wright's equation.

For notational simplicity we only derive the relation between $f(\mathbf{x})$ and $\tilde{W}$ for binary alleles. Let $\alpha = (\alpha_1, \ldots, \alpha_n)$ with $\alpha_i \in \{0, 1\}$ be a multi-index. We define with $0^0 := 1$:

$$\mathbf{x}^\alpha := \prod_i x_i^{\alpha_i}$$

Definition 2.1. *The representation of a binary discrete function using the ordering according to function values is given by*

$$f(\mathbf{x}) = f(0, \ldots, 0)(1 - x_1) \cdots (1 - x_n) + \cdots + f(1, \ldots, 1)x_1 \cdots x_n$$

The representation using the ordering according to variables is

$$f(\mathbf{x}) = \sum_\alpha a_\alpha x^\alpha \tag{2.10}$$

$\max\{|\alpha|_1 = \sum_i \alpha_i : \; a_\alpha \neq 0\}$ *is called the order of the function.*

In both representations the function is linear in each variable x_i. The following lemma is obvious.

Lemma 2.1. *The two representations are unique. There exist a unique matrix A of dimension $2^n * 2^n$ such that*

$$a_\alpha = (Af)_\alpha$$

We now use this result for $\tilde{W}$.

Lemma 2.2. $\tilde{W}(\mathbf{p}) := \bar{f}(t)$ *is an extension of $f(x)$ to S. There exist* *two representations for $\tilde{W}(p)$. These are given by*

(2.11)
$$\tilde{W}(\mathbf{p}) = f(0,\ldots,0)(1-p_1)\cdots(1-p_n) + \cdots + f(1,\ldots,1)p_1\cdots p_n$$

(2.12)
$$\tilde{W}(\mathbf{p}) = \sum_\alpha a_\alpha p^\alpha$$

The proofs in this section have been informal. The above lemma can rigorously be proven by Moebius inversion. If the function is given in analytical form (Equation 2.10) and the order of the function is bounded by a constant independent of n, $\tilde{W}(\mathbf{p})$ can be computed in polynomial time. The equation can also be used to compute the derivative of $\tilde{W}$, which is needed for Wright's equation. It is given by

(2.13)
$$\frac{\partial \tilde{W}(p)}{\partial p_i(1)} = \sum_{\alpha|\alpha_i=1} a_\alpha p^{\alpha'}$$

with $\alpha'_i = 0, \alpha'_j = \alpha_j$.
We will now characterize the attractors of UMDA. Let $S_i = \{q_i | \sum_{k\in\{0,1\}} q_i(x_k) \le 1; \quad 0 \le q_i(x_k) \le 1\}$ and $S = \prod_i S_i$ the Cartesian product. Then $S = [0,1]^n$ is the unit cube.

Theorem 2.3. *The stable attractors of Wright's equation are at the corners of S, i.e $p_i \in \{0,1\} \quad i = 1,\ldots,n$. In the interior there are only saddle points or local minima where grad $W(p)) = 0$. The attractors are local maxima of $f(x)$ according to one bit changes. Wright's equation solves the continuous optimization problem* argmax$\{\tilde{W}(\mathbf{p})\}$ *in S by gradient ascent.*

Proof: W is linear in p_i, therefore it cannot have any local maxima in the interior. Points with *grad* $W(p) = 0$ are unstable fixpoints of UMDA.

We next show that boundary points which are not local maxima of $f(x)$ cannot be attractors. We prove the conjecture indirectly. Without loss of generality, let the boundary point be $\hat{p} = (1,\ldots,1)$. We now consider an arbitrary neighbor, i.e $p^* = (0,1,\ldots,1)$. The two points are connected at the boundary by

$$p(z) = (1-z, 1,\ldots,1) \qquad z \in [0,1]$$

We know that $\tilde{W}$ is *linear* in the parameters p_i. Because $\tilde{W}(p^*) = f(0,1,\ldots,1)$ and $\tilde{W}(\hat{p}) = f(1,\ldots,1)$ we have

(2.14)
$$\tilde{W}(p(z)) = f(1,\ldots,1) + z\cdot\left[f(0,1,\ldots,1) - f(1,\ldots,1)\right].$$

If $f(0,1,\ldots,1) > f(1,\ldots,1)$ then $\hat{p}$ cannot be an attractor of UMDA. The mean fitness increases with z. $\square$

The extension of the above lemma to multiple alleles and multivariate distributions is straightforward, but the notation becomes difficult.

3 The Science of Breeding

Fitness proportionate selection is the undisputed selection method in population genetics. It is considered to be a model for *natural selection.* But for proportionate selection the following problem arises. When the population approaches an optimum, selection gets weaker and weaker because the fitness values become similar. Therefore breeders of livestock use other selection methods. These are called *artificial selection.* For large populations they mainly apply *truncation selection.* It works as follows. A truncation threshold τ is fixed. Then the τN best individuals are selected as parents for the next generation. These parents are then randomly mated.

The science of breeding is the domain of *quantitative genetics.* The theory is based on macroscopic variables. Because an exact mathematical analysis is impossible, many statistical techniques are used. In fact, the concepts of regression, correlation, heritability and decomposition of variance have been developed and applied in quantitative genetics for the first time.

3.1 Single Trait Theory

For a single trait the theory can be easily summarized. Starting with the fitness distribution, the *selection differential* $S(t)$ is introduced. It is the difference between the average of the selected parents and the average of the population.

$$S(t) = W(\mathbf{p}^s(t+1)) - W(\mathbf{p}(t)) \tag{3.1}$$

Similarly the response $R(t)$ is defined

$$R(t) = W(\mathbf{p}(t+1)) - W(\mathbf{p}(t)) \tag{3.2}$$

Next a linear regression is done

$$R(t) = b(t)S(t) \tag{3.3}$$

$b(t)$ is called *realized heritability.* The most difficult part of applying the theory is to predict $b(t)$. The first estimate uses the *regression of offspring to parent.* 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 3.1. *Let $P(t) = (f_1, \dots, f_N)$ be the population at generation t, where f_i denotes the phenotypic value of individual i. Assume that an offspring generation $O(t)$ is created by random mating, without selection. If the regression equation*

$$o_{ij}(t) = a(t) + b_{\bar{P}O}(t) \cdot \frac{f_i + f_j}{2} + \epsilon_{ij} \tag{3.4}$$

with

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

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

$$b_{\bar{P}O}(t) \approx b(t) \tag{3.5}$$

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

$$E(O(t)) = a(t) + b_{\bar{P}O}(t)M(t)$$

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

$$E(O(t)) = M(t)$$

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 still use the regression equation and obtain for the averages

$$M(t+1) = a(t) + b_{\bar{P}O}(t) \cdot M_s(t)$$

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

$$M(t+1) - M(t) = b_{\bar{P}O}(t) \cdot (M_s(t) - M(t))$$

This proves $b_{\bar{P}O}(t) = b(t)$.

The importance of regression for estimating the heritability was discovered by Galton and Pearson at the end of the 19th century. They computed the regression coefficient rather intuitively by scatter diagrams of mid-parent and offspring (Freedman *et al.*, 1991). The problem of computing a good regression coefficient is mathematically solved by the theorem of Gauss-Markov. We just cite the theorem. The proof can be found in any textbook on statistics (Rao, 1973).

Theorem 3.2. *A good estimate for the regression coefficient of mid-parent and offspring is given by*

$$b_{\bar{P}O}(t) = \frac{cov(O(t), \bar{P}(t))}{var(\bar{P}(t))} \tag{3.6}$$

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

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

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

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

The concept of covariance is restricted to parents producing offspring. It cannot be used for UMDA. Here the *analysis of variance* helps. We will decompose the fitness value $f(\mathbf{x})$ recursively into an additive part and interaction parts. We recall the definition of conditional probability.

Definition 3.1. *Let $p(\mathbf{x})$ denote the probability of $\mathbf{x}$. Then the conditional probability $p(\mathbf{x}|y)$ of $\mathbf{x}$ given y is defined by*

$$p(\mathbf{x}|y) = \frac{p(\mathbf{x}, y)}{p(y)} \tag{3.7}$$

First we extract the average.

$$f(\mathbf{x}) = \bar{f} + r_0(\mathbf{x}) \tag{3.8}$$

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

$$r_0(\mathbf{x}) = \sum_{i=1}^{n} f_{(i)}(x_i) + r_1(\mathbf{x}) \tag{3.9}$$

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

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

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 $r_1(\mathbf{x})$:

$$r_1(\mathbf{x}) = \sum_{\substack{(i,j)\\ i<j}} f_{(i,j)}(x_i, x_j) + r_2(\mathbf{x}) \tag{3.10}$$

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

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

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

$$V_k = \sum_{\substack{(i_i,\dots,i_k)\\ i_1<\dots<i_k}} \sum_{x_{i_1},\dots,x_{i_k}} p(x_{i_1}, \dots, x_{i_k}) f_{(i_i,\dots,i_k)}(x_{i_i}, \dots, x_{i_k})^2, \tag{3.11}$$

and

$$V_n = \sum_{\mathbf{x}} p(\mathbf{x}) r_{n-1}(\mathbf{x})^2 \tag{3.12}$$

If the population is in linkage equilibrium, the reader easily verifies that V_1 is the *additive genetic variance* defined by Equation 2.6. $p(x_{i_1}, \dots, x_{i_k})$ is a marginal probability distribution defined by $p(\mathbf{x})$. We are now able to formulate the theorem.

Theorem 3.3. *Let the population be in linkage equilibrium i.e.*

$$p(\mathbf{x}) = \prod_{i=1}^{n} p_i(x_i) \tag{3.13}$$

Then the variance of the population is given by

$$V = V_1 + V_2 + \cdots + V_{n-1} + V_n \tag{3.14}$$

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

$$cov(\bar{P}, O) = \frac{1}{2} V_1 + \frac{1}{4} V_2 + \cdots + \frac{1}{2^n} V_n = \sum_{k=1}^{n} \frac{1}{2^k} V_k \tag{3.15}$$

The proof can be found in (Asoh & Mühlenbein, 1994a). We now compare the estimates for heritability. For proportionate selection we have from Theorem 2.1

$$R_{UMDA}(t) = \frac{V_A(t)}{V(t)} S(t) + error_1(t).$$

For Two-Parent-Recombination (TPR) Mühlenbein (1997) has shown for $n = 2$ loci

$$R_{TPR}(t) = 2\frac{cov(\bar{P}(t), O(t))}{V(t)} S(t) + \frac{1}{2} error_2(t)$$

If the population is in linkage equilibrium we have $error_1 = error_2$ Using the covariance decomposition we can write

$$R_{TPR}(t) = \frac{VA(t)}{V(t)} S(t) + \frac{1}{2}\frac{V_2(t)}{V(t)} S(t) + \frac{1}{2} error(t)$$

Thus the first term of the expansion is identical to the $UMDA$ term. This shows again the similarity between two parent recombination and the $UMDA$ method. Breeders usually use the expression $b(t) = VA(t)/V(t)$ as an estimate. It is called *heritability in the narrow sense* (Falconer, 1981). But note that the variance decomposition seems to be only true for Robbins' proportions.

The selection differential is not suited for mathematical analysis. For truncation selection it can be approximated by

(3.16) $$S(t) \approx I_\tau V^{\frac{1}{2}}(t)$$

where I_τ is called the *selection intensity.* Combining the two equations we obtain the *famous equation for the response to selection.*

(3.17) $$R(t) = b(t) I_\tau V^{\frac{1}{2}}(t)$$

These equations are in depth discussed in (Mühlenbein, 1997). The theory of breeding uses macroscopic variables, the average and the variance of the population. But we have derived only one equation, the response to selection equation. We need a second equation connecting the average fitness and the variance in order to be able to compute the time evolution of the average fitness and the variance. There have been many attempts in population genetics to find a second equation. But all equations assume that the variance of the population continuously decreases. This is not the case for arbitrary fitness functions. Recently Prügel-Bennet and Shapiro (1997) have independently proposed to use moments for describing genetic algorithms. They apply methods of statistical physics to derive equations for higher moments for special fitness functions.

3.2 Tournament Selection

Another important scheme is *tournament selection of size k*. Here k individuals are randomly chosen. The best individual is taken as parent. We model binary tournament selection ($k = 2$) as a game. Two individuals with genotype $\mathbf{x}$ and $\mathbf{y}$ "play" against each other. The one with the larger fitness gets a payoff of 2. If the fitness values are equal, both will win half of the games. This gives a payoff of 1. The game is defined by a *payoff matrix* with coefficients

$$a_{xy} = \begin{cases} 2 & f(\mathbf{x}) > f(\mathbf{y}) \\ 1 & f(\mathbf{x}) = f(\mathbf{y}) \\ 0 & f(\mathbf{x}) < f(\mathbf{y}) \end{cases}$$

With some effort one can show that

$$\sum_{\mathbf{x}} \sum_{\mathbf{y}} p(\mathbf{x}, t) a_{xy} p(\mathbf{y}, t) = 1 \tag{3.18}$$

After a round of tournaments the genotype frequencies are given by

$$p^s(\mathbf{x}, t+1) = p(\mathbf{x}, t) \sum_{\mathbf{y}} a_{xy} p(\mathbf{y}, t). \tag{3.19}$$

If we set

$$b(\mathbf{x}, t) = \sum_{\mathbf{y}} a_{xy} p(\mathbf{y}, t),$$

then the above equation is similar to proportionate selection using the function $b(\mathbf{x}, t)$. But b depends on the genotype frequencies. Furthermore the average $\bar{b}(t) = \sum p(\mathbf{x}, t) b(\mathbf{x}, t)$ remains constant, $\bar{b}(t) \equiv 1$.

The difference equations for the univariate marginal frequencies can be derived in the same manner as for proportionate selection. They are given by

$$p_i(x_i, t+1) = p(x_i, t) \cdot \bar{B}_i(t) \tag{3.20}$$

$$\bar{B}_i(t) = \sum_{\mathbf{x}, X_i = x_i} b(\mathbf{x}, t) \prod_{\substack{j=1 \\ j \neq i}}^{n} p_j(x_j, t) \tag{3.21}$$

The difference equation for binary tournament selection is more difficult than for proportionate selection. $\bar{B}_i$ is quadratic in $p(x_j)$. The fitness value of $\mathbf{x}$ is given by $\sum_y a_{xy} p(\mathbf{y}, t)$. This is called called *frequency dependent fitness* in population genetics.

Tournament selection uses only the order relation of the fitness values. The fitness values themselves do not change the outcome of a tournament. Therefore the evolution of the univariate marginal frequencies depends on the order relation only. The same is true for truncation selection. Thus tournament selection can be approximated by truncation selection. For each k there exists a selection intensity I_k with

$$S(t) = I_k V^{\frac{1}{2}}(t)$$

For $k = 2$ we have $I_2 = 1/\sqrt{\pi} = 0.564$ ((Mühlenbein, 1997)).

3.3 Analytical Results for Linear Functions

For the special case that all univariate marginal distributions are equal, i.e. $p_i := p$, it is possible to obtain an analytical solution for $p(t)$.

We cite from (Mühlenbein, 1997) the analytical solutions for the linear function $OneMax(n) = \sum_i x_i$. For completeness we give the difference equation and its solution.

Theorem 3.4. *If in the initial population all univariate marginal frequencies are identical to $p_0 > 0$, then we obtain for UMDA and OneMax proportionate selection:*

$$R(t) = 1 - p(t) \tag{3.22}$$

$$p(t) = 1 - (1 - p_0)(1 - \frac{1}{n})^t \tag{3.23}$$

truncation selection:

$$R(t) \approx I_\tau \sqrt{np(t)(1 - p(t))} \tag{3.24}$$

$$p(t) \approx 0.5 \left(1 + \sin\left(\frac{I_\tau}{\sqrt{n}}t + \arcsin(2p_0 - 1)\right)\right) \tag{3.25}$$

tournament selection:

(3.26)
$$R(t) = np(1-p)\Big(2\sum_{k=1}^{n}\sum_{j=0}^{k-1}\binom{n-1}{k-1}\binom{n}{j}p^{k+j-1}(1-p)^{2n-k-j-1} + \sum_{k=1}^{n-1}\binom{n-1}{k-1}\binom{n}{k}p^{2k-1}(1-p)^{2n-2k-1} - \sum_{j=0}^{2n-2}p^j\Big)$$

(3.27)
$$R(t) \approx 0.564\sqrt{np(t)(1 - p(t))}$$

The formulas can be used to compute the number of generations until convergence (GEN_e). For truncation selection convergence is defined by $p(t) = 1$, for proportionate selection by $p(t) = 1 - \epsilon$.

Corollary 3.1. *The number of generations until convergence is given by:*

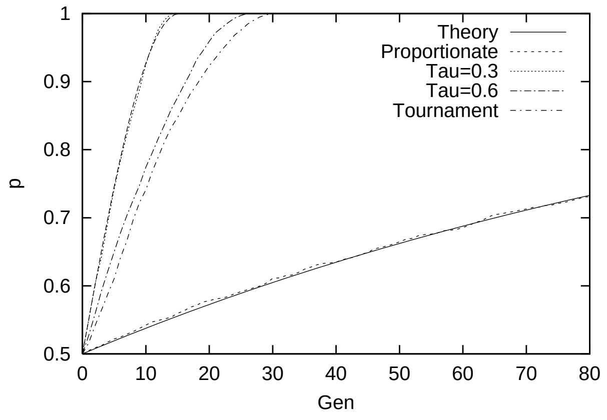

FIGURE 3.1. Comparison of selection methods for OneMax(128)

proportionate selection:

$$GEN_e = n \cdot ln \frac{1 - p_0}{\epsilon} \tag{3.28}$$

truncation selection:

$$GEN_e = \left(\frac{\pi}{2} - \arcsin(2p_0 - 1)\right) \frac{\sqrt{n}}{I_\tau} \tag{3.29}$$

Truncation selection converges in $O(\sqrt{n})$ and proportionate selection in $O(-n \cdot ln(\epsilon))$ generations. Numerical results have shown that truncation selection converges in about $O\sqrt{n}$ till $O(n)$ generations (Mühlenbein & Mahnig, 2000) for all fitness functions optimized.

The analytical solutions almost perfectly match the results obtained from actual UMDA runs (see figure 3.1). With proportionate selection the population needs a long time to approach the optimum. In contrast, truncation selection and tournament selection lead to much faster convergence. p increases almost linearly until near the optimum. Equation 3.26 for binary tournament selection has p^{2n} as the largest exponent. This complicated equation can be approximated by Equation 3.27 with surprising accuracy.

We next present numerical results for some popular fitness functions.

3.4 Numerical Results for UMDA

This section solves the problem put forward by Mitchell et al. (1994): to understand the class of problems for which genetic algorithms are most suited, and in particular, for which they will outperform other

Table 3.1
Mean function evaluations for Royal Road(8). U is UMDA, F FDA

1+1	SGA	U: p	U: $\tau = 0.3$	U: $\tau = 0.05$	F: $\tau = 0.3$
6,334	61,334	55,586	28,000	14,264	7,634

search algorithm. We start with the *Royal Road* function, which was erroneously believed to lay out a royal road for the GA to follow to the optimal string.

3.5 Royal Road Function

We discuss the Royal Road function R_1, which was used by Mitchell et al. (1994). It is defined as follows:

$$R_1(l,x) = \sum_{i=0}^{l-1} \prod_{j=1}^{8} x_{8i+j} \tag{3.30}$$

The function is of order 8. The Building Block Hypothesis BBH (Holland, 1975/1992) states that "the GA works well when instances of low-order, short schemas that confer high fitness can be recombined to form instances of larger schemas that confer even higher fitness." In our terminology a schema defines a marginal distribution. Thus a first-order schema defines a univariate marginal distribution. Our analysis has shown that only the first half of the BBH is correct: first order schemata of high fitness are recombined. Larger schemata play no role.

Table 3.1 confirms and extends the results of Mitchell et al. (1994). The really bad performance of SGA is mainly a result of proportionate selection. UMDA with proportionate selection (U: p) needs slightly less evaluations. With very strong selection, UMDA needs only about twice as much function evaluations as the $(1+1)$-algorithm. This algorithm performs a random bit flip and accepts a new configuration if its fitness is equal or better. The good performance of this algorithm has already been shown in (Mühlenbein, 1991). But it performs only good if the fitness function never decreases with increasing number of bits. Almost identical performance to the $(1+1)$-algorithm can be obtained by FDA. It uses marginal distributions of size 8 instead of univariate marginal distributions. It will be explained in Section 4.3.

Figure 3.2 shows once more the importance of selection. Proportionate selection performs very good in the beginning, because the fitness values of all strings containing no building block are zero. These strings are not reproduced. But after 5 generations proportionate selection gets weaker. Truncation selection with $\tau = 0.3$ overtakes it after 23 generations. We just mention, that the numerical results would be much worse for pro-

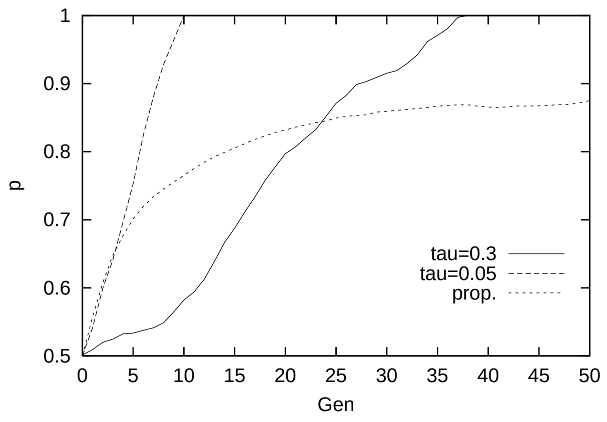

FIGURE 3.2. Convergence of Royal Road

portionate selection, if we add 1 to the Royal Road function. In this case proportionate selection selects also many strings without a building block.

We will now explain the results by using our theory to analytically solve the equations. We have

$$\begin{aligned}\tilde{W}(\mathbf{p}) &= \sum_{i=0}^{l-1}\prod_{j=1}^{8} p_{8i+j}\\ \frac{\partial\tilde{W}}{\partial p_k} &= \prod_{\substack{j=1\\8i+j\neq k}}^{7} p_{8i+j} \qquad 8i \le k < 8i+8\end{aligned}$$

For truncation selection we will apply the response to selection equation. Therefore we have to compute the variance $V_l(t)$. We simplify the computation by observing that the blocks of 8 variables are independent and therefore

$$V_l(t) = l \cdot V_1(t).$$

We recall that all function values are 0 except $f(1,\dots,1)$. Therefore

$$\begin{aligned}V_1(t) &= \sum_x p(x,t)f^2(x) - W^2\\ &= \prod p_i - (\prod p_i)^2\end{aligned}$$

If we assume that $p_i = p$ for all i we obtain

(3.31) $$V_8(t) = 8p(t)^8(1-p(t)^8)$$

We can now formulate the theorem.

Theorem 3.5. *If all univariate marginal distributions are identical to $p(t)$ and $p(0) = p_0$ then we obtain for proportionate selection*

$$p(t+1) - p(t) = \frac{1 - p(t)}{8} \tag{3.32}$$

$$p(t) = 1 - (1 - p_0)(\tfrac{7}{8})^t \tag{3.33}$$

For truncation selection with threshold τ we approximately get

$$R(t) \approx b(t) I_\tau \sqrt{8p(t)^8(1 - p(t)^8)} \tag{3.34}$$

$$p(t)^8 \approx 0.5\left(1 + \sin\Big(\frac{b(t)I_\tau}{\sqrt{8}}t + \arcsin(2p_0^8 - 1)\Big)\right) \tag{3.35}$$

Proof: The conjectures for proportionate selection directly follow from Equation 2.4. From the response to selection equation we obtain

$$8p(t+1)^8 - 8p(t)^8 \approx b(t) I_\tau \sqrt{8p(t)^8(1 - p(t)^8)}$$

If we set $q(t) = p(t)^8$ the above equation is identical to the equation for $OneMax(8)$. The approximate solution is given by Equation 3.25. $\square$

In order to apply Equation 3.35 we need an estimate for the realized heritability $b(t)$. Experiments show that $b(t)$ increases approximately linearly from about 0 to 1. Thus we set $b(t) \propto t$. A numerical comparison between Equation 3.35 and a simulation with truncation threshold 0.05 shows only 5% difference. The coincidence between theory and simulation is very good.
This example shows that the response to selection equation can in special cases be used to compute an analytical solution for $p(t)$. The difficulty is to determine the heritability $b(t)$.

3.6 Multi-modal Functions Suited for UMDA Optimization

Equation 2.4 shows that UMDA performs a gradient ascent in the landscape given by W. This helps our search for functions best suited for UMDA. We take the *Saw* landscape as a spectacular example. The definition of the function can be extrapolated from Figure 3.3. In Saw(n, m, k), n denotes the number of bits and $2m$ the distance from one peak to the next. The highest peak is multiplied by k (with $k \leq 1$), the second highest by k^2, then k^3 and so on. The landscape is very rugged. In order to get from one local optimum to another one, one has to cross a deep valley.

But again the transformed landscape $W(\mathbf{p})$ is fairly smooth. An example is shown in Figure 3.4. Whereas $f(\mathbf{x})$ has 5 isolated peaks, $W(\mathbf{p})$ has three plateaus, a local peak and the global peak. We will use *UMDA*

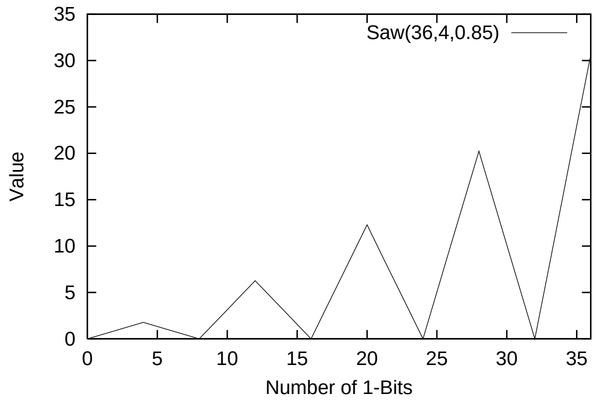

FIGURE 3.3. Definition of Saw(36,4,0.85)

with truncation selection. We have not been able to derive precise analytical expressions. In Figure 3.4 the results are displayed.
In the simulation two truncation thresholds, $\tau = 0.05$ and $\tau = 0.01$, have been used. For $\tau = 0.05$ the probability p stops at the local maximum for $\tilde{W}(\mathbf{p})$. It is approximately $p = 0.78$. For $\tau = 0.01$ $UMDA$ is able to converge to the optimum $p = 1$. It does so by even going downhill!

This example confirms in a nutshell our theory. *$UMDA$ transforms the original fitness landscape defined by $f(\mathbf{x})$ into a fitness landscape defined by $\tilde{W}(\mathbf{p})$. This transformation smoothes the rugged fitness landscape $f(\mathbf{x})$. $UMDA$ might find the global optimum, if there is a tendency towards the global optimum.*

This example shows that UMDA can solve difficult multi-modal optimization problems. It is obvious that any search method using a single search point like the $(1+1)$-algorithm needs an almost exponential number of function evaluations.

3.7 Deceptive Functions

There exist many optimization problems where $UMDA$ is mislead. $UMDA$ will converge to local optima, because it does not use correlations between the variables. We demonstrate this problem by a deceptive function. We use the definition

$$\text{Decep}(\mathbf{x}, k) := \begin{cases} k - 1 - |\mathbf{x}|_1 & 0 \leq |\mathbf{x}|_1 < k \\ k & |\mathbf{x}|_1 = k \end{cases} \tag{3.36}$$

The global maximum is isolated at $x = (1, \dots 1)$. A deceptive function

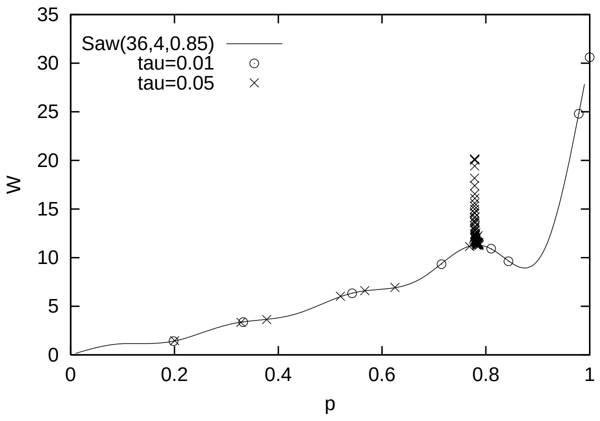

Figure 3.4. Results with normal and strong selection.

of order n is a needle in a haystack problem. This is far too difficult to optimize for any optimization method. We simplify the optimization problem by adding l distinct $Decep(k)$-functions to give a fitness function of size $n = l * k$. This function is also deceptive. The local optimum $x = (0, \ldots, 0)$ is surrounded by good fitness values, whereas the global optimum is isolated.

$$\text{Decep}(n,k) = \sum_{i=1,k+1,\ldots}^{n} \text{Decep}\big((x_i, x_{i+1}, \ldots, x_{i+k-1}), k\big) \tag{3.37}$$

Our theory easily shows that at $p_i = 0.5$ the gradient points to $x_i = 0$. Thus starting at $p(0) = 0.5$ $UMDA$ converges to the local optimum $\mathbf{x} = (0, \ldots, 0)$. This problem can be solved, if higher order marginal distributions are used. This will be discussed later in the context of the Factorized Distribution Algorithm FDA.

We next show how the science of breeding can be used for controlling $UMDA$.

3.8 Numerical Investigations of the Science of Breeding

The application of the science of breeding needs the computation of the average fitness $\bar{f}(t)$, the variance $V(t)$ and the additive genetic variance $VA(t)$. The first two terms are standard statistical terms. The computation of VA needs $\bar{f}_i(x_i)$ and $p_i(x_i)$. The computation of the first term

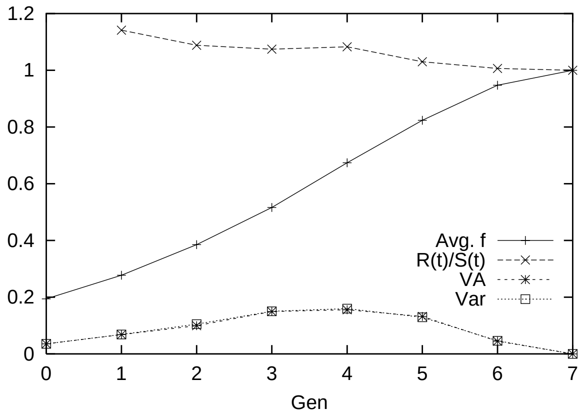

FIGURE 3.5. Heritability and Variance for a multiplicative function (variance and VA multiplied by 10); $s = 0.1$, $n = 32$

only poses some difficulties. It can be approximated by

$$\bar{f}_i(X_i = 1, t) = \sum_{x, X_i = 1} \frac{p(\mathbf{x})}{p_i(X_i = 1)} f(\mathbf{x}) \approx \frac{1}{N} \sum_{k=1}^{N} \frac{f(\zeta_i^k)}{p_i(x_i)} \tag{3.38}$$

ζ_i^k are those $\mathbf{x}$ values in the population which contain $x_i = 1$.

Linear functions are the ideal case for the theory. The heritability $b(t)$ is 1 and the additive genetic variance is identical to the variance. We skip this trivial case and start with a multiplicative fitness function $f(\mathbf{x}) = \prod_i (1-s)^{1-x_i}$. For a multiplicative function we also have $R(t) = S(t)$ (Theorem 2.2).

Figure 3.5 confirms the theoretical results from Section 1 (VA and Var are multiplied by 10 in this figure). Additive genetic variance is almost identical to the variance and the heritability is 1. The function is highly nonlinear of order n, but nevertheless it is easy to optimize. The function has also been investigated by Rattray and Shapiro (1999). But their calculations are very difficult.

An interesting case is the function $Deceptive - 4$. In Figure 3.6 the function is optimized for 32 bits. As predicted by the theory $UMDA$ converges to the local optimum $\mathbf{x} = (0, \ldots, 0)$. Heritability is almost zero at the beginning, indicating that the competition between setting the genes to 0 or to 1 is undecided. $UMDA$ decides to go to the direction of 0. If there is a high percentage of zeros in the population, then heritability increases to almost 1. In this area the fitness function is almost linear.

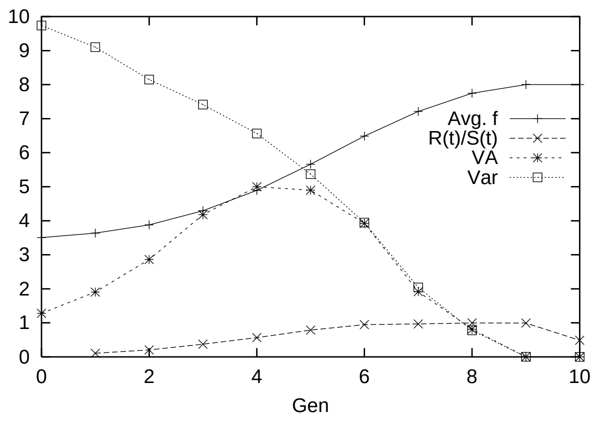

FIGURE 3.6. Heritability and Variance for Dec-4: Average, Var and VA divided by 3; $\tau = 0.3$, $n = 32$

This shows that heritability can strongly depend on the gene frequencies.

The examples demonstrate that it is worthwhile to compute the quantities used for a scientific breeding program. They clearly indicate how difficult the optimization problem is. In breeding of livestock heritability is normally greater than than 0.2. If we optimize arbitrary fitness functions the heritability can be almost 0. But because we can easily compute 1000 generations on a computer in a few minutes, $UMDA$ can be used for problems with very low heritability.

We have shown that $UMDA$ can optimize difficult multi-modal functions, thus explaining the success of genetic algorithms in optimization. We have also shown that $UMDA$ can easily be deceived by simple functions called deceptive functions. These functions need more complex search distributions. This problem is investigated next.

4 Graphical Models and Optimization

The simple product distribution of $UMDA$ cannot capture dependencies between variables. If these dependencies are necessary to find the global optimum, $UMDA$ and simple genetic algorithms fail. We take an extreme case as example, the *needle in a haystack problem.* The fitness function is everywhere one, except for a single $\mathbf{x}$ where it is 10. All x_i values have to be set in the right order to obtain the optimum. Of course, there exist no clever search method for this problem. But there is a continuum of increasing complexity from the simple *OneMax* function to the needle in a haystack. For complex problems we need a complex search distribution. A good candidate for a search distribution for optimization is the Boltzmann distribution.

Definition 4.1. *For $\beta \geq 0$ define the* weighted Boltzmann distribution *of a function $f(\mathbf{x})$ as*

$$p_{\beta,f}(\mathbf{x}) := \frac{p_0(\mathbf{x})e^{\beta f(\mathbf{x})}}{\sum_y p_0(\mathbf{y})e^{\beta f(\mathbf{y})}} := \frac{p_0(\mathbf{x})e^{\beta f(\mathbf{x})}}{Z_f(\beta, p_0)} \tag{4.1}$$

where $Z_f(\beta, p_0)$ is the partition function. To simplify the notation β and/or f can be omitted. $p_0(\mathbf{x})$ is the distribution for $\beta = 0$.

The Boltzmann distribution concentrates the search around good fitness values. Thus it is theoretically a very good candidate for a search distribution used for optimization. The problem lies in the efficient computation of the Boltzmann distribution. The theory presented in this section unifies simulated annealing and population based algorithms with the general theory of estimating distributions.

4.1 The Factorized Distribution Algorithm FDA

The Boltzmann distribution is usually defined as $e^{-\frac{g(\mathbf{x})}{T}}/Z$. The term $g(\mathbf{x})$ is called the energy and $T = 1/\beta$ the temperature. The weighted Boltzmann distribution has a number of properties, among them

Lemma 4.1. *Let $x_m \in \mathcal{M}$ be a global optimum of the function $f(\mathbf{x})$ and $\mathbf{x}_l$ a point with $f(\mathbf{x}_l) < f(\mathbf{x}_m)$. Then*

- *Let $g(\mathbf{x}) := f(\mathbf{x}) + c$. Then $p_{\beta,f}(\mathbf{x}) = p_{\beta,g}(\mathbf{x})$.*
- *Let $g(\mathbf{x}) := c \cdot f(\mathbf{x})$. Then $p_{\beta,g}(\mathbf{x}) = p_{c\beta,f}(\mathbf{x})$.*

The first property means that the distribution is invariant under addition of a constant. It is, however, not invariant under multiplication. We will discuss how to overcome this shortcoming in Section 4.4.

The Boltzmann distribution is a suitable distribution for optimization because it concentrates its weight with increasing β around the global optima of the function. If it was possible to sample efficiently from this distribution for arbitrary β, optimization would be almost trivial.

4.2 Boltzmann selection

Closely related to the Boltzmann distribution is Boltzmann selection. An early study about this selection method can be found in (de la Maza & Tidor, 1993).

Definition 4.2. *Given a distribution p and a selection parameter γ, **Boltzmann selection** calculates the distribution of the selected points according to*

$$p^s(\mathbf{x}) = \frac{p(\mathbf{x})e^{\gamma f(\mathbf{x})}}{\sum_y p(\mathbf{y})e^{\gamma f(\mathbf{y})}} \tag{4.2}$$

This allows us to define the *BEDA* (Boltzmann Estimated Distribution Algorithm).

BEDA – Boltzmann Estimated Distribution Algorithm

- **STEP 0:** $t \Leftarrow 0$. Generate N points according to the $p(\mathbf{x}, 0) = p_0(\mathbf{x})$.
- **STEP 1:** With a given $\Delta\beta(t) > 0$, let
$$p^s(\mathbf{x}, t) = \frac{p(\mathbf{x}, t)e^{\Delta\beta(t)f(\mathbf{x})}}{\sum_y p(\mathbf{y}, t)e^{\Delta\beta(t)f(\mathbf{y})}}.$$
- **STEP 2:** Generate N new points according to the distribution $p(x, t+1) = p^s(x, t)$.
- **STEP 3:** $t \Leftarrow t + 1$.
- **STEP 4:** If stopping criterion not met go to STEP 1

BEDA is a conceptional algorithm, because the calculation of the distribution requires to compute the sum of exponentially many terms. The following convergence theorem is easily proven.

Theorem 4.1 (Convergence). *Let $\Delta\beta(t)$ be an annealing schedule, i.e. for every t increase the inverse temperature β by $\Delta\beta(t)$. Then for BEDA the distribution at time t is given by*

$$p(\mathbf{x}, t) = \frac{p_0(\mathbf{x})e^{\beta(t)f(\mathbf{x})}}{Z_f(\beta(t), p_0)} \tag{4.3}$$

with the inverse temperature

$$\beta(t) = \sum_{\tau=1}^{t} \Delta\beta(\tau). \tag{4.4}$$

Let $\mathcal{M}$ be the set of global optima. If $\beta(t) \to \infty$, then

$$\lim_{t\to\infty} p(x, t) = \begin{cases} 1/|\mathcal{M}| & x \in \mathcal{M} \\ 0 & else \end{cases} \tag{4.5}$$

Proof: Let $x^m \in \mathcal{M}$ be a point with optimal fitness and $x \notin \mathcal{M}$ a point with $f(\mathbf{x}) < f(x^m)$. Then

$$p(x,t) = \frac{p_0(\mathbf{x})e^{\beta(t)f(\mathbf{x})}}{\sum_y p_0(y)e^{\beta(t)f(y)}} \leq \frac{e^{\beta(t)f(\mathbf{x})}}{|\mathcal{M}| \cdot C \cdot e^{\beta(t)f(x^m)}}$$
$$\leq \frac{1}{|\mathcal{M}| \cdot C \cdot e^{\beta(t)[f(x^m)-f(\mathbf{x})]}}$$

As $\beta(t) \to \infty$, $p(x,t)$ converges (exponentially fast) to 0. Because $p(x,t) = p(y,t)$ for all $x^m, y^m \in \mathcal{M}$, the limit distribution is the uniform distribution on the set of optima. $\square$

Equation (4.5) only shows that the distribution converges to 0 for non-optimal points. But we can also make an estimate for the rate of convergence:

Lemma 4.2. *Let $p_0(\mathbf{x})$ be the uniform distribution. Let there be a δ such that for any non-optimal point x we have with $x^m \in \mathcal{M}$*

$$f(\mathbf{x}) \leq f(x^m) - \delta \tag{4.6}$$

Then

$$\beta \geq \frac{n \cdot \ln 2}{\delta} \quad \Longrightarrow \quad p_\beta(\mathcal{M}) \geq 0.5. \tag{4.7}$$

Proof: Let $|\mathcal{M}|$ be the number of optima. The number of terms in the partition function is smaller than 2^n. For $x^m \in \mathcal{M}$ we have with $M := f(x^m)$

$$p_\beta(x^m) = \frac{e^{\beta M}}{\sum_y e^{\beta f(y)}} \geq \frac{e^{\beta M}}{2^n \cdot e^{\beta(M-\delta)} + |\mathcal{M}| \cdot e^{\beta M}}$$
$$= \frac{1}{e^{n\ln 2 - \beta\delta} + |\mathcal{M}|} \quad \overset{!}{\geq} \frac{1}{2|\mathcal{M}|}, \tag{4.8}$$

So, to have $p_\beta(\mathcal{M}) \geq 1/2$, we need

$$e^{n \ln 2 - \beta\delta} \leq 2|\mathcal{M}| \iff \beta \geq \frac{n \cdot 2 - \ln(2|\mathcal{M}|)}{\delta} \tag{4.9}$$

or as a sufficient condition (4.7). $\square$

Corollary 4.1. *For a binary fitness function with integer values half of the generated points will have maximum fitness if $\beta \geq 0.7n$, independent of the fitness function.*

We next transform $BEDA$ into a practical algorithm. This means the reduction of the parameters of the distribution and the computation of an adaptive schedule.

4.3 Factorization of the distribution

In this section we describe a method for computing a factorization of the probability, given an additive decomposition of the function:

Definition 4.3. *Let $s_1, \ldots, s_m$ be index sets, $s_i \subseteq \{1, \ldots, n\}$. Let f_{s_i} be functions depending only on the variables x_j with $j \in s_i$. These variables we denote as x_{s_i} Then*

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_{s_i}(\mathbf{x}) = f_i(x_{s_i}) \tag{4.10}$$

is an additive decomposition *of the fitness function f.*

We also need the following definitions

Definition 4.4. *Given $s_1, \ldots, s_m$, we define for $i = 1, \ldots, m$ the sets d_i, b_i and c_i:*

$$d_i := \bigcup_{j=1}^{i} s_j, \qquad b_i := s_i \setminus d_{i-1}, \qquad c_i := s_i \cap d_{i-1} \tag{4.11}$$

We set $d_0 = \emptyset$.

In the theory of decomposable graphs, d_i are called *histories*, b_i *residuals* and c_i *separators* (Lauritzen, 1996). We recall the following definition.

Definition 4.5. *The conditional probability $p(\mathbf{x}|\mathbf{y})$ is defined as*

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})} \tag{4.12}$$

In (Mühlenbein *et al.*, 1999), we have shown the following theorem.

Theorem 4.2 (Factorization Theorem). *Let $p(\mathbf{x})$ be a Boltzmann distribution with*

$$p(\mathbf{x}) = \frac{e^{\beta f(\mathbf{x})}}{Z_f(\beta)} \tag{4.13}$$

and $f(\mathbf{x}) = \sum_{i=1}^{m} f_{s_i}(\mathbf{x})$ be an additive decomposition. If

$$b_i \neq \emptyset \quad \forall i = 1, \ldots, l; \quad d_l = \tilde{X}, \tag{4.14}$$

$$\forall i \geq 2 \, \exists j < i \text{ such that } c_i \subseteq s_j \tag{4.15}$$

then

$$p(\mathbf{x}) = \prod\nolimits_{i=1}^{m} p(x_{b_i}|x_{c_i}) \tag{4.16}$$

The constraint defined by Equation (4.15) is called the *running intersection property* (Lauritzen, 1996).

FDA – Factorized Distribution Algorithm

- **STEP 0:** Calculate b_i and c_i from the decomposition of the function.
- **STEP 1:** Generate an initial population with N individuals.
- **STEP 2:** Select N individuals using Boltzmann selection.
- **STEP 3:** Estimate the conditional probabilities $p(x_{b_i}|x_{c_i}, t)$ from the selected points.
- **STEP 4:** Generate new points according to $p(\mathbf{x}, t+1) = \prod_{i=1}^{m} p(x_{b_i}|x_{c_i}, t)$.
- **STEP 5:** If not stopping criterion reached: $t \Leftarrow t+1$ Go To STEP2

With the help of the factorization theorem, we can turn the conceptional algorithm $BEDA$ into FDA, the Factorized Distribution Algorithm. As the factorized distribution is identical to the Boltzmann distribution if the conditions of the factorization theorem are fulfilled, the convergence proof of $BEDA$ also applies to FDA.

Not every additive decomposition leads to a factorization using the factorization theorem. In these cases, more sophisticated methods have to be used. But FDA can also be used with an approximate factorization.

We discuss two simple examples.

Example 4.1. *For linear functions*

$$Linear(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i x_i \tag{4.17}$$

we have $s_i = \{i\}$ and thus all c_i are empty. This leads to the factorization

$$p(\mathbf{x}) = \prod_{i=1}^{n} p_i(x_i). \tag{4.18}$$

As this is the distribution used by $UMDA$, FDA behaves like $UMDA$ (and thus like a simple genetic algorithm) for linear functions.

Example 4.2. *Functions with a chain-like interaction can also be factorized:*

$$Chain(\mathbf{x}) = \sum_{i=2}^{n} f_i(x_{i-1}, x_i) \tag{4.19}$$

Here the factorization is

$$p(\mathbf{x}) = p(x_1) \prod_{i=2}^{n} p(x_i|x_{i-1}) \tag{4.20}$$

FDA can be used with any selection scheme, but then the convergence proof is no longer valid. We think that Boltzmann selection is an essential part in using the *FDA*. In order to obtain a practical algorithm, we still have to solve two problems: To find a good annealing schedule for Boltzmann selection and to determine a reasonable sample size (population size).

These two problems will be investigated next.

4.4 A new annealing schedule for the Boltzmann distribution

Boltzmann selection needs an annealing schedule. Lemma 4.2 has shown how fast we have to anneal in order to reach convergence within a given time frame. But if we anneal too fast, the approximation of the Boltzmann due to the sampling error can be very bad. For an extreme case, if the annealing parameter is very large, the second generation should consist only of the global maxima.

4.4.1 Taylor expansion of the average fitness

In order to determine an adaptive annealing schedule, we will make a Taylor expansion of the average fitness of the Boltzmann distribution.

Definition 4.6. *The* ***average fitness*** *of a fitness function and a distribution is*

$$W_f(p) = \sum_x f(\mathbf{x})p(\mathbf{x}) \tag{4.21}$$

For the Boltzmann distribution, we use the abbreviation $W_f(\beta) := W_f(p_{\beta,f})$.

Theorem 4.3. *The average fitness of the Boltzmann distribution* $W_f(\beta)$ *has the following expansion in* β*:*

$$W_f(\tilde{\beta}) = W_f(\beta) + \sum_{i \geq 1} \frac{(\tilde{\beta} - \beta)^i}{i\,!} M^c_{i+1}(\beta) \tag{4.22}$$

where M^c_i *are the centred moments*

$$M^c_i(\beta) := \sum_x \left[f(\mathbf{x}) - W_f(\beta)\right]^i p(\mathbf{x}) \tag{4.23}$$

They can be calculated using the derivatives of the partition function:

$$M^c_{i+1}(\beta) = \left(\frac{Z'_f(\beta)}{Z_f(\beta)}\right)^{(i)} \qquad \text{for } i \geq 1, \quad M^c_1 = 0 \tag{4.24}$$

Proof: The k-th derivative of the partition function obeys for $k \geq 0$:

$$Z^{(k)}_f(\beta) = \sum_x f(\mathbf{x})^k e^{\beta f(\mathbf{x})} \tag{4.25}$$

Thus the moments for $k \geq 1$ can be calculated as

$$M_k(\beta) := \sum_x f(\mathbf{x})^k p(\mathbf{x}) = \frac{Z_f^{(k)}(\beta)}{Z_f(\beta)} \tag{4.26}$$

and thus

$$W_f(\beta) = M_1(\beta) = Z_f'(\beta)/Z_f(\beta). \tag{4.27}$$

Direct evaluation of the derivatives of W leads to complicate expressions. The proof is rather technical by induction. We omit it here. □

Corollary 4.2. *We have approximative*

$$W_f(\tilde{\beta}) \approx W_f(\beta) + (\tilde{\beta} - \beta) \cdot \sigma_f^2(\beta) \tag{4.28}$$

where $\sigma_f^2(\beta)$ *is the variance of the distribution, defined as* $\sigma_f^2(\beta) := M_2^c(\beta)$.

This approximation can also be found in (Kirkpatrick *et al.*, 1983).

Lemma 4.3. *The variance of the Boltzmann distribution obeys*

$$f(\mathbf{x}) \neq const. \implies \sigma_f^2(\beta) > 0 \tag{4.29}$$

Proof: We have $\forall x : p_\beta(\mathbf{x}) > 0$. In order to have

$$\sigma_f^2(\beta) = \sum_x \left[f(\mathbf{x}) - W_f(\beta)\right]^2 p_\beta(\mathbf{x}) \overset{!}{=} 0, \tag{4.30}$$

we must have for all x: $f(\mathbf{x}) = W_f$ in contradiction to the assumption. □

Corollary 4.3. *With* $f(\mathbf{x}) \neq konst.$ *we have*

$$\tilde{\beta} > \beta \implies W_f(\tilde{\beta}) > W_f(\beta) \tag{4.31}$$

The corollary shows that the average fitness never decreases for Boltzmann selection. A similar result was already obtained in Theorem 2.1 for proportional selection (see also (Mühlenbein & Mahnig, 2000)).

4.4.2 The SDS Annealing Schedule

From (4.28) we can derive an adaptive annealing schedule. The variance (and the higher moments) can be estimated from the generated points. As long as the approximation is valid, one can choose a desired increase in the average fitness and set $\beta(t+1)$ accordingly. So we can set

$$\Delta\beta(t) := \beta(t+1) - \beta(t) = \frac{W_f^{\text{new}}(t) - W_f(\beta(t))}{\sigma_f^2(\beta(t))} \tag{4.32}$$

From (4.28) we see that choosing $\Delta\beta$ proportional to the inverse of the variance leads in the approximation to a constant increase in the average fitness. This is much too fast, especially near the optimum. As truncation selection has proven to be a robust and efficient selection scheme, we can try to approximate the behaviour of this method. For truncation selection the *response to selection* $R_f(t)$ is approximatively given by equation 3.17

$$R(t) := W((t+1) - W(t) \approx I_\tau b(t)\sqrt{\sigma_f^2} \tag{4.33}$$

I_τ is the selection intensity, depending on the truncation threshold τ. Because truncation selection has been shown to be an effective selection method, we will make the Boltzmann schedule proportional to the inverse of the square root of the variance:

Definition 4.7. *The standard deviation schedule (SDS) is defined by*

$$\Delta\beta(t) = \frac{c}{\sigma_f(\beta(t))} \tag{4.34}$$

We already know that FDA with Boltzmann selection remains unchanged if we add a constant to the fitness function. For SDS we have additionally

Lemma 4.4. *For Boltzmann selection with SDS, $BEDA$ is invariant under linear transformations of the fitness function with a positive factor.*

Proof: This lemma is true because the standard deviation scales linearly under multiplication. Let $f(\mathbf{x})$ be a fitness function, consider $\hat{f}(\mathbf{x}) = \hat{c} \cdot f(\mathbf{x})$. The claim is that $\hat{\beta}(t) = \beta(t)/\tilde{c}$, then the distributions are the same for every t. With $t{=}0$, β and $\hat{\beta}$ are 0, so it is true. Let now t and $\beta{=}\beta(t)$ be given. From the previous iteration we know that $\hat{\beta} = \beta/\hat{c}$.

According to lemma 4.1, we have $p_{\beta,f}(\mathbf{x}){=}p_{\hat{\beta},\hat{f}}(\mathbf{x})$. Also, $\sigma_{\hat{f}}^2(\beta) = \hat{c}^2 \cdot \sigma_{\hat{f}}^2(\hat{\beta})$. Hence we have $\Delta\hat{\beta}(t){=}\Delta\beta(t)/\hat{c}$. □

Corollary 4.4. *Let $\sigma(t)$ be the standard deviation. Then the response to selection for Boltzmann selection with the SDS is given by*

$$R_f(t) = \sum_{i\geq 1} \frac{c^i}{i!\,\sigma(t)^i} M_{i+1}^c \tag{4.35}$$

$$= c \cdot \sigma(t) + \frac{c^2 M_3^c}{2\,\sigma(t)^2} + \frac{c^3 M_4^c}{6\,\sigma(t)^3} + \dots \tag{4.36}$$

Note that this annealing schedule cannot be used for simulated annealing, as the estimation of the variance of the distribution requires samples that are independently drawn. But the sequence of samples generated by simulated annealing are not independent.

4.4.3 Linear functions

For linear functions

$$Linear(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i x_i \tag{4.37}$$

the factorization of the Boltzmann distribution was calculated in equation (4.18). We can also calculate the partition function and get

$$Z_f(\beta) = \prod_{i=1}^{n} (1 + e^{\beta\alpha_i}) \tag{4.38}$$

and

$$p_i(\beta) := p_\beta(X_i = 1) = \frac{e^{\beta\alpha_i}}{1 + e^{\beta\alpha_i}}. \tag{4.39}$$

Because of the independence of the variables, the variance is just the sum of the variance of the factors and we have

$$\sigma_f^2(\beta) = \sum_{i=1}^{n} \frac{\alpha_i^2 e^{\beta\alpha_i}}{(1 + e^{\beta\alpha_i})^2} = \sum_{i=1}^{n} \alpha_i^2 p_i(\beta)\bigl(1 - p_i(\beta)\bigr) \tag{4.40}$$

and thus

$$\beta(t+1) = \beta(t) + \frac{c}{\sqrt{\sum_i \alpha_i^2 p_i(\beta)\bigl(1 - p_i(\beta)\bigr)}} \tag{4.41}$$

By differentiating (4.39) we get

$$\begin{aligned} \frac{dp_i(\beta)}{dt} &= \frac{\alpha_i e^{\beta\alpha_i}(1 + e^{\beta\alpha_i})\beta' - e^{\beta\alpha_i}\alpha_i e^{\beta\alpha_i}\beta'}{(1 + e^{\beta\alpha_i})^2} \\ &= p_i(\beta)\bigl(1 - p_i(\beta)\bigr)\alpha_i \tfrac{d\beta}{dt} \end{aligned} \tag{4.42}$$

Therefore we obtain the differential equation

$$\frac{dp_i(\beta)}{dt} = c \cdot \frac{p_i(\beta)\bigl(1 - p_i(\beta)\bigr)\alpha_i}{\sqrt{\sum_i \alpha_i^2 p_i(\beta)\bigl(1 - p_i(\beta)\bigr)}} \tag{4.43}$$

Note that the solution of these differential equations remain the same if we multiply all α_i by a constant factor, as predicted.

For *Onemax* we have $\alpha_i = 1$. In this case all marginal frequencies are equal to p_β. We obtain the differential equation

$$\frac{dp_\beta}{dt} = c\sqrt{p_\beta(1 - p_\beta)/n} \tag{4.44}$$

The solution of this equation is given by Equation 3.25.
We next turn to the fixation problem in finite populations.

4.5 Finite Populations

In finite populations convergence of $UMDA$ or FDA can only be probabilistic. Since $UMDA$ a simple FDA algorithm, it is sufficient to discuss FDA. This section is extracted from (Mühlenbein & Mahnig, 1999b).

Definition 4.8. *Let ϵ be given. Let $P_{conv}(N)$ denote the probability that FDA with a population size of N converges to the optima. Then the critical population size is defined as*

$$N^*(\epsilon) = \min_N P_{conv}(N) \geq 1 - \epsilon \tag{4.45}$$

If FDA with a finite population does not convergence to an optimum, then at least one gene is fixed to a wrong value. The probability of fixation is reduced if the population size is increased. We obviously have for FDA

$$P_{conv}(N_1) \leq P_{conv}(N_2) \quad N_1 \leq N_2$$

The critical question is: How many sample points are necessary to reasonably approximate the distribution used by FDA. A general estimate from Vapnik (Vapnik, 1998) can be a guideline. One should use a sample size which is about 20 times larger than the number of free parameters.

We discuss the problem with a special function called *Int*. *Int*($\mathbf{x}$) gives the integer value of the binary representation.

$$Int(n) = \sum_{i=1}^{n} 2^{i-1} x_i \tag{4.46}$$

The fitness distribution of this function is not normal distributed. The function has 2^n different fitness values. We show the cumulative fixation probability in Table 4.1 for *Int*(16). The fixation probability is larger for stronger selection. For a given truncation selection the maximum fixation probability is at generation 1 for very small N. For larger values of N the fixation probability increases until a maximum is reached and then decreases again. This behaviour has been observed for many fitness distributions.

For truncation selection with $\tau = 0.25$ we have for $N = 80$ a fixation probability of about 0.075. A larger τ reduces the fixation probability. But this advantage is set off by the larger number of generations needed to converge. The problem of an optimal population size for truncation selection is investigated in (Mühlenbein & Mahnig, 1999b). Boltzmann selection with $\Delta\beta = 0.01$ is still very strong for the fitness distribution given by *Int*(16). For $N = 700$ the largest fixation probability is still at the first generation. Therefore the critical population size for Boltzmann selection for $\Delta\beta = 0.01$ is very high ($N^* > 700$). In comparison, the adaptive Boltzmann schedule SDS has a total fixation probability of

TABLE 4.1
Cumulative fixation probability for *Int*(16). Truncation selection vs. Boltzmann selection with $\Delta\beta = 0.01$ and Boltzmann SDS; N denotes size of population.

t	$\tau = 0.25$ $N = 30$	$\tau = 0.5$ $N = 30$	$\tau = 0.25$ $N = 80$	$\tau = 0.5$ $N = 60$	*Boltz.* $N = 700$	*SDS* $N = 100$
1	0.0955	0.0035	0.0	0.0	0.0885	0.0
2	0.4065	0.0255	0.0025	0.0095	0.1110	0.0
3	0.5955	0.1040	0.0165	0.0205	0.1275	0.0
4	0.6880	0.2220	0.0355	0.0325	0.1375	0.002
5	0.7210	0.3270	0.0575	0.0490	0.1455	0.002
6	0.7310	0.4030	0.0695	0.0630	0.1510	0.008
7	0.7310	0.4470	0.0740	0.0715	0.1555	0.018
8	0.7310	0.4705	0.0740	0.0780	0.1565	0.030
9	0.7310	0.4840	0.0740	0.0806	0.1575	0.036
14						0.084

0.084 for a population size of $N = 100$. This is almost as small as truncation selection.
This example shows that Boltzmann selection in finite populations critically depends on a good annealing schedule. Normally we run FDA with truncation selection. This selection method is a good compromise. But Boltzmann selection with SDS schedule is of comparable performance.

Estimates for the necessary size of the population can also be found in (Harik *et al.*, 1999). But they use a weaker performance definition. The goal is to have a certain percentage of the bits of the optimum in the final population. Furthermore their result is only valid for fitness function which are approximately normally distributed.

The danger of fixation can further be reduced by a technique very popular in Bayesian statistics. This is discussed in the next section.

4.6 Population Size, Mutation, and Bayesian Prior

In order to derive the results of this section we will use a normalized representation of the distribution.

Theorem 4.4 (Bayesian Factorization). *Each probability can be factored into*

$$p(\mathbf{x}) = p(x_1) \prod_{i=2}^{n} p(x_i|pa_i) \tag{4.47}$$

Proof: By definition of conditional probabilities we have

$$p(\mathbf{x}) = p(x_1)\prod_{i=2}^{n} p(x_i|x_1,\cdots,x_{i-1}) \tag{4.48}$$

Let $pa_i \subset \{x_1,\cdots,x_{i-1}\}$. If x_i and $\{x_1,\cdots,x_{i-1}\}\setminus pa_i$ are conditionally independent given pa_i, we can simplify $p(x_i|x_1,\cdots,x_{i-1}) = p(x_i|pa_i)$. $\square$

PA_i are called the parents of variable X_i. This factorization can be represented by a directed graph. In the context of graphical models the graph and the conditional probabilities are called a *Bayesian network* (Jordan, 1999; Frey, 1998). It is obvious that the factorization used in Theorem 4.2 can be easily transformed into a Bayesian factorization.

Usually the empirical probabilities are computed by the maximum likelihood estimator. For N samples with $m \le N$ instances of x the estimate is defined by

$$\hat{p}(x) = \frac{m}{N}$$

For $m = N$ we obtain $p(x) = 1$ and for $m = 0$ we obtain $p(x) = 0$. This leads to our gene fixation problem, because both values are attractors. The fixation problem is reduced if $\hat{p}(x)$ is restricted to an interval $0 < p_{min} \le \hat{p}(x) \le 1 - p_{min} < 1$. This is exactly what results from the *Bayesian estimation*. The estimate $\hat{p}(x)$ is the expected value of the posterior distribution after applying Bayes formula to a prior distribution and the given data. For binary variables x the estimate

$$\hat{p}(x) = \frac{m+r}{N+2r} \tag{4.49}$$

is used with $r > 0$. r is derived from a Bayesian prior. $r = 1$ is the result of the uniform Bayesian prior. The larger r, the more the estimates tend towards 1/2. The reader interested in a derivation of this estimate in the context of Bayesian networks is referred to (Jordan, 1999).

The Bayesian prior can be seen as a mutation force. Wright ((1970)) included mutation with a recurrent symmetric mutation rate of $0 \le \mu < 1$ as follows into his equation

$$\Delta p_i = p_i(t)(1-p_i(t))\frac{\frac{\partial \tilde{W}}{\partial p_i}}{\tilde{W}} - \mu\big(p_i(t) - (1-p_i(t))\big) \tag{4.50}$$

A similar equation is obtained if a Bayesian prior is used. We use the formula

$$p_i(t+1) = \frac{p_i^s(t)N + r}{N+2r}$$

where $p_i^s(t)$ is given by Wright's equation 2.4. Setting $\gamma = r/N$ we obtain

$$\Delta p_i(t) = p_i(t) + p_i(t)(1-p_i(t))\tfrac{\partial \tilde{W}}{\tilde{W}} + \frac{\gamma}{1+2\gamma} - \frac{2\gamma}{1+2\gamma}\left(p_i(t) + p_i(t)(1-p_i(t))\tfrac{\partial \tilde{W}}{\tilde{W}}\right) \tag{4.51}$$

The attractors of this equation are in the interior of the unit cube. The location is given by an equilibrium between selection and mutation. How can we determine an appropriate value for r for our FDA application? In principle should the location be as far as possible in the interior under te constraint that the optima are generated with high probability. In this paper we make a simplified analysis.

The uniform prior gives for $m = 0$ the value $\hat{p}_{min} = 1/(N+2)$. If N is small, then p_{min} might be so large that we generate the optima with a very small probability only. This means we perform more a random search instead of converging to the optima. This consideration leads to a constraint. $1 - p_{min}$ should be so large that the optima are still generated with high probability. We now heuristically derive p_{min} under the assumption that there is a unique optimum. To simplify the formulas we require that $\max \hat{p}(x_{opt}) \geq e^{-1}$.

This means that the optimum string x_{opt} should be generated more than 30% at equilibrium. This is large enough to observe equilibrium and convergence. Let us first investigate the $UMDA$ factorization $p(x) = \prod p(x_i)$. For $r = 1$ the largest probability is $p_{max} = (N+1)/(N+2)$. Obviously

$$p_{max} = 1 - \frac{1}{N+2} = 1 - p_{min}$$

The largest probability to generate the optimum is given by

$$\hat{p}(x_{opt}) = \prod_{i=1}^{n}(1 - \frac{1}{N+2}) \approx e^{-\frac{n}{N+2}}$$

If $N = O(n^{1-\alpha})$ with $\alpha > 0$, then $p(x_{opt})$ becomes arbitrarily small for large n. For $N = n$ we obtain $\hat{p}(x_{opt}) \approx e^{-1}$. This gives the following guideline, which actually is a lower bound of the population size.

Rule of Thumb: *For $UMDA$ the size of the population should be at least equal to the size of the problem, if a Bayesian prior of $r = 1$ is used.*

Bayesian priors are also defined for conditional distributions. The above heuristic derivation can also be used for general Bayesian factorizations. The Bayesian estimator is for binary variables

$$\hat{p}(x_i|pa_i) = \frac{m+r}{P+2r}$$

P is the number of occurrences of pa_i. We make the assumption that in the best case the optimum constitutes 25% of the population. This gives $P \geq N/4$. For $r = 1$ we compute as before

$$\hat{p}(x_{opt}) = \prod_{i=1}^{n} \hat{p}(x_{opt_i}|pa_{opt_i}) = \prod_{i=1}^{n}(1 - \frac{1}{N/4+2}) \approx e^{-\frac{n}{N/4+2}}$$

If we set $N = 4n$ we obtain $\hat{p}(x_{opt}) \approx e^{-1}$. Thus we obtain a lower bound for the population size:

Rule of Thumb: *For FDA using a factorization with many conditional distributions and Bayesian prior of $r = 1$, the size of the population should be about four times the size of the problem.*

These rule of thumbs have been heuristically derived. They have to be confirmed by numerical studies. Our FDA estimate is a crude lower bound. There exist more general estimates. We just cite Vapnik (Vapnik, 1998). In order to approximate a distribution with a reasonable accuracy, he proposes to use a sample size which is about 20 times larger than the number of free parameters of the distribution. For $UMDA$ this gives $20n$, i.e. 20 times our estimate.

We demonstrate the importance of using a Bayesian prior by an example. It is a deceptive function of order 4 and problem size of $n = 32$. Our convergence theorem gives convergence of FDA with Boltzmann selection and an exact factorization. The exact factorization consists of marginal distributions of size 4. We compare in Figure 4.1 FDA with SDS Boltzmann selection and truncation selection without Bayesian prior. We also show a run with SDS Boltzman selection and Bayesian prior.

The simulation was started at $p = 0.15$, i.e. near the local optimum $p = 0$. Nevertheless, FDA converges to the global optimum at $p = 1$. It is interesting to note that FDA at first moves into the direction of the local optimum. At the very last moment the direction of the curve is dramatically changed. SDS Boltzmann selection behaves almost identical to truncation selection with threshold $\tau = 0.35$. But both methods need a huge population size in order to converge to the optimum. In this example it is $N = 20000$. If a prior of $r = 1$ is used the population size can be reduced to $N = 200$. With this prior the curve changes direction earlier. Because of the prior the univariate marginal probabilities never reach $p = 0$ or $p = 1$. In this example p stops at about $p = 0.975$.

Let us now summarize the results: Because FDA uses finite samples

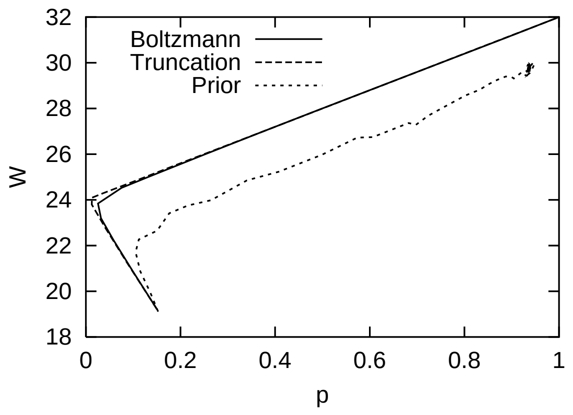

Figure 4.1. Average fitness $W(p)$ for FDA for Decep(32,4); population size $N = 20000$ without prior and $N = 200$ with prior $r = 1$.

of points to estimate the conditional probabilities, convergence to the optimum will depend on the size of the samples (the population size). FDA has experimentally proven to be very successful on a number of functions where standard genetic algorithms fail to find the global optimum. In (Mühlenbein & Mahnig, 1999b), the scaling behaviour for various test functions has been studied. The estimation of the probabilities and the generation of new points can be done in polynomial time. If a Bayesian prior is used the influence of the population size is reduced. There is a tradeoff. If no prior is used then convergence is fast. But a large population size might be needed. If a prior is used, the population size can be much smaller. But the number of generations until convergence increases. We have not yet enough numerical results, therefore we just conjecture:
FDA with a finite population of size $N = 4n$, SDS Boltzmann selection, Bayesian prior, and a Bayesian factorization where the number of parents is restricted by k independent of n, will converge to the optimum in polynomial time with high probability.

4.7 Constraint Optimization Problems

An advantage of FDA compared to genetic algorithm is that it can handle optimization problems with constraints. Mendelian recombination or crossover in genetic algorithms often creates points which violate the constraints. If the structure of the constraints and the structure of the ADF are compatible, then FDA will generate only legal points.

Definition 4.9. *A constraint optimization problem is defined by*

$$maxf(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}_{s_i}) \tag{4.52}$$

$$s.t. C_i(\mathbf{x}_{u_i}) \tag{4.53}$$

$C_i(\mathbf{x}_{u_i})$ stands for the ith constraint function. $\mathbf{x}_{s_i}, \mathbf{x}_{u_i} \subseteq X$ are sets of variables. The constraints are locally defined. Thus they can be used to test which marginal probabilities are 0. This is technically somewhat complicated, but straightforward. For instance, if we have $C_1(x_1, x_2) = x_1 + x_2 \leq 1$, then obviously $p(X_1 = 1, X_2 = 1) = 0$. Thus the constraints are mapped to marginal distributions: if $C_i(\mathbf{x}_{u_i})$ is violated then we set $p_i(x_{u_i}) = 0$.

We can now factorize $f(\mathbf{x})$ as before. But we can also factorize the graph defined by $C_i(x_{u_i})$. Our theory can handle the two cases: the factorization of the constraints is contained in the factorization of the function, i.e. $x_{u_i} \subseteq x_{s_i}$, or the factorization of the function is contained in the factorization of the constraints, i.e. $x_{s_i} \subseteq x_{u_i}$

Let Ω_c be the set of *feasible* solutions. Then the Boltzmann distribution on Ω_c is defined as

$$p_{b,f,c}(\mathbf{x}) = \frac{p_0(x)e^{\beta f(\mathbf{x})}}{\sum_{y \in \Omega_c} p_0(\mathbf{y}) e^{\beta f(\mathbf{y})}} \tag{4.54}$$

Then the following convergence theorem holds.

Theorem 4.5 (Convergence). *Let 1) the initial population be feasible. Let 2) the factorization of thetechniques. constraints and the factorization of the function be contained in the FDA factorization. Let 3) $\Delta\beta(t)$ be an annealing schedule. Then for FDA the distribution at time t is given by*

$$p(\mathbf{x}, t) = \frac{p_0(\mathbf{x})e^{\beta(t) f(\mathbf{x})}}{\sum_{y \in \Omega_c} p_0(\mathbf{y}) e^{\beta(t) f(y)}} \tag{4.55}$$

with the inverse temperature

$$\beta(t) = \sum_{\tau=1}^{t} \Delta\beta(\tau). \tag{4.56}$$

Let $\mathcal{M}$ be the set of global optima. If $\beta(t) \to \infty$, then

$$\lim_{t \to \infty} p(x, t) = \begin{cases} 1/|\mathcal{M}| & x \in \mathcal{M} \\ 0 & else \end{cases} \tag{4.57}$$

Proof: The proof is almost identical to the proof of Theorem 4.1. We only have to show that the factorization generates feasible solutions only, if the probabilities are computed from a set of feasible solutions. The proof is indirect. Suppose there exists $\mathbf{x}$ which does not satisfy the kth constrain $C_k(\mathbf{x}_{u_k})$. Then

$$0 \neq p(\mathbf{x}, t+1) = \prod_{i=1}^{n} p^s(\mathbf{x}_{b_i}|\mathbf{x}_{c_i}, t)$$

Thus we have $p^s(\mathbf{x}_{u_k}) \neq 0$. This means that there exist at least one individual in generation t which violates the constraint. But this is a contradiction to assumption 1). □

The factorization theorem needs an analytical description of the function. But it is also possible to determine the factorization from the data sampled. This is described next.

5 Computing a Bayesian Network from Data

The FDA factorization is based on the decomposition of the fitness function. This has two drawbacks: first, the structure of the function has to be known. Second, for a given instance of the fitness function, the structure might not give the smallest factorization possible. In other words: complex structures are not necessarily connected to corresponding complex dependency structures for a given fitness function. The actual dependencies depend on the actual function values. This problem can be circumvented by computing the dependency structure from the data.

Computing the structure of a Bayesian network from data is called learning. Learning gives an answer to the question: *Given a population of selected points $M(t)$, what is a good Bayesian factorization fitting the data?* The most difficult part of the problem is to define a quality measure also called scoring measure.

A Bayesian network with more arcs fits the data better than one with less arcs. Therefore a scoring metric should give the best score to the minimal Bayesian network which fits the data. It is outside the scope of this paper to discuss this problem in more detail. The interested reader is referred to the two papers by Heckerman and Friedman et al. in (Jordan, 1999).

For Bayesian networks two quality measures are most frequently used - the *Bayes Dirichlet* (BDe) score and the *Minimal Description Length* (*MDL*) score. We concentrate on the *MDL* principle. This principle is motivated by universal coding. Suppose we are given a set D of instances, which we would like to store. Naturally, we would like to conserve space and save a compressed version of D. One way of compressing the data is to find a suitable model for D that the encoder can use to produce a compact version of D. In order to recover D we must also store the model used by the encoder to compress D. Thus the total description length is defined as the sum of the length of the compressed version of D and the length of the description of the model. The *MDL* principle postulates that the optimal model is the one that minimizes the total description length.

5.1 LFDA - Learning a Bayesian Factorization

In the context of learning Bayesian networks, the model is a network B describing a probability distribution p over the instances appearing in the data. Several authors have approximately computed the *MDL* score. Let $M = |D|$ denote the size of the data set. Then *MDL* is approximately given by

$$MDL(B,D) = -\operatorname{ld}(P(B)) + M \cdot H(B,D) + \tfrac{1}{2} PA \cdot \operatorname{ld}(M) \tag{5.1}$$

with $\operatorname{ld}(x) := \log_2(x)$. $P(B)$ denotes the prior probability of network B, $PA = \sum_i 2^{|pa_i|}$ gives the total number of probabilities to compute. $H(B,D)$ is defined by

$$H(B,D) = -\sum_{i=1}^{n} \sum_{pa_i} \sum_{x_i} \frac{m(x_i,pa_i)}{M} \operatorname{ld} \frac{m(x_i,pa_i)}{m(pa_i)} \tag{5.2}$$

where $m(x_i,pa_i)$ denotes the number of occurrences of x_i given configuration pa_i. $m(pa_i) = \sum_{x_i} m(x_i,pa_i)$. If $pa_i = \emptyset$, then $m(x_i,\emptyset)$ is set to the number of occurrences of x_i in D.

The formula has an interpretation which can be easily understood. If no prior information is available, $P(B)$ is identical for all possible networks. For minimizing, this term can be left out. $0.5PA \cdot \operatorname{ld}(M)$ is the length required to code the parameter of the model with precision 1/M. Normally one would need $PA \cdot \operatorname{ld}(M)$ bits to encode the parameters. However, the central limit theorem says that these frequencies are roughly normally distributed with a variance of $M^{-1/2}$. Hence, the higher $0.5\operatorname{ld}(M)$ bits are not very useful and can be left out. $-M \cdot H(B,D)$ has two interpretations. First, it is identical to the logarithm of the maximum likelihood ($\operatorname{ld}(L(B|D))$). Thus we arrive at the following principle:

Choose the model which maximizes $\operatorname{ld}(L(B|D)) - \frac{1}{2} PA \cdot \operatorname{ld}(M)$.

The second interpretation arises from the observation that H(B,D) is the conditional entropy of the network structure B, defined by PA_i, and the data D. The above principle is appealing, because it has no parameter to be tuned. But the formula has been derived under many simplifications. In practice, one needs more control about the quality vs. complexity tradeoff. Therefore we use a weight factor α. Our measure is defined by BIC.

$$BIC(B, D, \alpha) = -M \cdot H(B, D) - \alpha PA \cdot \mathrm{ld}(M) \tag{5.3}$$

This measure with $\alpha = 0.5$ has been first derived by Schwarz (1978) as *Bayesian Information Criterion.* Therefore we abbreviate our measure as $BIC(\alpha)$.

To compute a network B^* which maximizes BIC requires a search through the space of all Bayesian networks. Such a search is more expensive than to search for the optima of the function. Therefore the following greedy algorithm has been used. k_{max} is the maximum number of incoming edges allowed.

BN(α, k$_{\mathbf{max}}$)

- **STEP 0:** Start with an arc-less network.
- **STEP 1:** Add the arc (x_i, x_j) which gives the maximum increase of BIC(α) if $|PA_j| \leq k_{max}$ and adding the arc does not introduce a cycle.
- **STEP 2:** Stop if no arc is found.

Checking whether an arc would introduce a cycle can be easily done by maintaining for each node a list of parents and ancestors, i.e. parents of parents etc. Then $(x_i \rightarrow x_j)$ introduces a cycle if x_j is ancestor of x_i.

The BOA algorithm of Pelikan (Pelikan *et al.*, 2000) uses the BDe score. This measure has the following drawback. It is more sensitive to coincidental correlations implied by the data than the *MDL* measure. As a consequence, the BDe measure will prefer network structures with more arcs over simpler networks (Bouckaert, 1994). The BIC measure with $\alpha = 1$ has also been proposed by Harik (1999). But Harik allows only factorizations without conditional distributions. This distribution is only correct for separable functions.

Given the BIC score we have several options to extend FDA to $LFDA$ which learns a factorization. Due to limitations of space we can only show results of an algorithm which computes a Bayesian network at each generation using algorithm $BN(0.5, k_{max})$. FDA and $LFDA$ should behave fairly similar, if $LFDA$ computes factorizations which are in probability terms very similar to the FDA factorization. FDA uses the same factorization for all generations, whereas $LFDA$ computes a new factorization at each step which depends on the given data M.

TABLE 5.1
Numerical results for different algorithms, $LFDA$ with $BN(\alpha, 8)$

Function	n	α	N	τ	Succ.%	SDev
OneMax	30	$UMDA$	30	0.3	75	4.3
	30	0.25	100	0.3	2	1.4
	30	0.5	100	0.3	38	4.9
	30	0.75	100	0.3	80	4.0
	30	0.25	200	0.3	71	4.5
Saw(32,2,0.5)	32	$UMDA$	50	0.5	71	4.5
	32	$UMDA$	200	0.5	100	0.0
	32	0.25	200	0.5	41	2.2
	32	0.5	200	0.5	83	1.7
	32	0.75	200	0.5	96	0.9
	32	0.25	400	0.5	84	3.7
Deceptive-4	32	$UMDA$	800	0.3	0	0.0
	32	FDA	100	0.3	81	3.9
	32	0.25	800	0.3	92	2.7
	32	0.5	800	0.3	72	4.5
	32	0.75	800	0.3	12	3.2

We have applied $LFDA$ to many problems (Mühlenbein & Mahnig, 1999b). The results are encouraging. Here we only discuss the functions introduced in Section 3.4. We recall that UMDA finds the optimum of the multi modal functions BigJump and Saw. $UMDA$ uses univariate marginal distributions only. Therefore its Bayesian network has no arcs.

Table 5.1 summarizes the results. For $LFDA$ we used three different values of α, namely $\alpha = 0.25, 0.5, 0.75$. The smaller α, the less penalty for the size of the structure. Let us discuss the results in more detail. $\alpha = 0.25$ gives by far the best results when a network with many arcs is needed. This is the case for Deceptive-4. Here a Bayesian network with three parents is optimal. $\alpha = 0.25$ performs bad on problems where a network with no arcs defines a good search distribution. For the linear function $OneMax$ $BIC(0.25)$ has only a success rate of 2%. The success rate can be improved if a larger population size N is used. The reason is as follows. $BIC(0.25)$ allows denser networks. But if a small population is used, spurious correlations may arise. These correlations have a negative impact for the search distribution. The problem can be solved by using a larger population. Increasing the value from $N = 100$ to $N = 200$ increases the success rate from 2% to 71% for OneMax.

For *Saw* a Bayesian network with no arcs is able to generate the optimum. An exact factorization requires a factor with n parameters. We used the heuristic BN with $k_{max} = 8$. Therefore the exact factorization cannot be found. In all these cases $\alpha = 0.75$ gives the best results. $BIC(0.75)$ enforces smaller networks. But $BIC(0.75)$ performs very bad on Deceptive-4. Taking all results together $BIC(0.5)$ gives good results. This numerical results supports the theoretical estimate.

The numerical result indicates that control of the weight factor α can substantially reduce the amount of computation. For Bayesian network we have not yet experimented with control strategies. We have intensively studied the problem in the context of neural networks (Zhang *et al.*, 1997).

UMDA most efficiently optimizes the functions $OneMax$ and Saw. FDA is efficient if the exact factorization needs a small number of parents in the Bayesian graph ($k \leq 5$). $LFDA$ most of the time also finds the optimum. From the functions considered it has the largest difficulty with the function Saw. The performance of $LFDA$ can be substantially improved, if for each fitness function a suitable value of α is chosen. We recall that a small value of α leads to more complex Bayesian factorizations. The BIC score uses $\alpha = 0.5$. This value is a good compromise. But $\alpha = 0.75$ gives much better performance for the functions $OneMax$ and Saw, whereas $\alpha = 0.25$ gives the best results for the function $Decep(36, 4)$. These results are explained next.

5.2 Optimization, Dependencies, and Search Distributions

We have proven in Section 4.2 convergence of FDA with Boltzmann selection to the set of global maxima. If the Boltzmann distribution can be factorized, the computational complexity for one generation is bounded by $O(n \cdot N \cdot 2^k)$. k denotes the maximum number of parents. A factorization can be determined if the fitness function is decomposed. It can also be obtained from the data sampled. Unfortunately for many interesting applications k is very large. If the fitness function is additively decomposed on a $2D$ grid of size n, then k scales like $O(\sqrt{n})$. It is easy to show that k scales even like $O(n)$ for the function Saw.

But we have demonstrated that the simple search distribution used by $UMDA$ guides the search to the optimum of Saw. The reason is that for Saw we have a tendency: the more bits on, the higher the fitness value. Therefore an exact Boltzmann factorization is not needed for optimization! The problem of finding a good approximation of the Boltzmann distribution which generates the optima with high probability cannot be solved theoretically. Therefore we propose the following heuristic.

Multi-Factorization LFDA *Use different values of α in order to* *obtain factorizations of different complexity. In a standard setting, use* $\alpha = 0.25$, 0.5, *and* 0.75. *Generate new search points using the different factorizations for a certain percentage of the population.*

6 The System Dynamics Approach to Optimization

We have shown that Wright's equations converge to some local optima of the fitness function at the boundary. We might ask ourselves: Why not using the difference equations directly, without generating a population? This approach is callled the systems dynamics approach to optimization. We just discuss a few examples which are connected with our theory.

6.1 The Replicator Equation

In this section we investigate the relation between Wright's equation and a popular equation called *replicator equation.* Replicator dynamics is a standard model in evolutionary biology to describe the dynamics of growth and decay of a number of species under selection. Let $S = \{1, 2, \ldots, s\}$ be a set of species, p_i the frequency of species i in a fixed population of size N. Then the replicator equation is defined on a simplex $S^s = \{p : \sum p_i = 1, 0 \leq p_i \leq 1\}$

$$\frac{dp_i}{dt} = p_i(t)\left(f_i(\mathbf{p}) - \sum_{i=1}^{s} p_i(t) f_i(\mathbf{p})\right) \tag{6.1}$$

f_i gives the fitness of species i in relation to the others. The replicator equation is discussed in detail in (Hofbauer & Sigmund, 1998). For the replicator equation a maximum principle can be shown.

Theorem 6.1. *If there exists a potential* V *with* $\partial V/\partial p_i = f_i(\mathbf{p})$, *then* $dV/dt \geq 0$, *i.e the potential* V *increases using the replicator dynamics.*

If we want to apply the replicator equation to a binary optimization problem of size n, we have to set $s = 2^n$. Thus the number of species is exponential in the size of the problem. The replicator equation can be used for small size problems only.

Voigt (1989) had the idea, to generalize the replicator equation by introducing continuous variables $0 \leq p_i(x_k) \leq 1$ with $\sum_k p_i(x_k) = 1$. Thus $p_i(x_k)$ can be interpreted as univariate probabilities. Voigt (1989) proposed the following discrete equation.

Definition 6.1. *The* Discrete Diversified Replicator Equation *DDRP* *is given by*

$$(6.2) \quad p_i(x_k)(t+1) - p_i(x_k)(t) = p_i(x_k)(t)\frac{f_{ik}(\mathbf{p}) - \sum_{x_k} p_i(x_k)f_{ik}(\mathbf{p})}{\sum_{x_k} p_i(x_k)f_{ik}(\mathbf{p})}$$

The name Discrete Diversified Replicator Equation was not a good choice. The DDRP is more similar to Wright's equation than to the replicator equation. This is the content of the next theorem.

Theorem 6.2. *If the average fitness $W(\mathbf{p})$ is used as potential, then Wright's equation and the Discrete Diversified Replicator Equation are identical.*

Proof: The average fitness is defines as

$$W(\mathbf{p}) = V(\mathbf{p}) = \sum_x a_x \prod_{i=1}^n p_i(x_i)$$

We compute the derivatives

$$\frac{\partial V(\mathbf{p})}{\partial p_i(1)} = \sum_{x|x_i=1} a_x \prod_{j\neq i}^n p_j(x_j)$$

$$\frac{\partial V(\mathbf{p})}{\partial p_i(0)} = \sum_{x|x_i=0} a_x \prod_{j\neq i}^n p_j(x_j)$$

Obviously

$$p_i(1)\frac{\partial V}{\partial p_i(1)} + p_i(1)\frac{\partial V}{\partial p_i(1)} = V(\mathbf{p})$$

The conjecture now follows from the proof of Wright's equation. □

We recently discovered that Baum and Eagon (1967) have proven a discrete maximum principle for certain instances of the DDRP.

Theorem 6.3 (Baum-Eagon). *Let $V(\mathbf{p})$ be a polynomial with non-negative coefficients homogeneous of degree d in its variables $p_i(x_j)$ with $p_i(x_j) \geq 0$ and $\sum_{x_j} p_i(x_j) = 1$. Let $\mathbf{p}(t+1)$ be the point given by*

$$(6.3) \qquad p_i(x_j, t+1) = \frac{p_i(x_j,t)\frac{\partial V}{\partial p_i(x_j)}}{\sum_{x_k} p_i(x_k)\frac{\partial V}{\partial p_i(x_k)}}$$

The derivatives are taken at $\mathbf{p}(t)$. Then $V(\mathbf{p}(t+1)) > V(\mathbf{p}(t)$ unless $\mathbf{p}(t+1) = \mathbf{p}(t)$

Equation 6.3 is exactly the DDRP with a potential V. Thus the DDRP could be called the Baum-Eagon equation. From the above theorem the discrete maximum principle for Wright's equation follows by setting $V = W$ and $d = n$. Thus the potential is the average fitness, which is homogeneous of degree n.

6.2 Some System Dynamics Equations for Optimization

Theorem 6.3 shows that both, Wright's equation and the DDRP, maximize some potential. This means that both equations can be used for maximization. But there is a problem: both equations are deterministic. For difficult optimization problems, there exists a large number of attractors, each with a corresponding attractor region. If the iteration starts at a point within the attractor region, it will converge to the corresponding attractor at the boundary. But if the iteration starts at points which lie at the boundary of two or more attractors, i.e on the separatrix, the iteration will be confined to the separatrix. The deterministic system cannot decide for one of the attractors.

$UMDA$ with a finite population does not have a sharp boundary between attractor regions. We model this behavior by introducing randomness. The new value $p_i(x_j, t+1)$ is randomly chosen from the interval

$$[(1-c)p_i'(x_j, t+1), (1+c)p_i'(x_j, t+1)]$$

$p_i'(x_j, t+1)$ is determined by the deterministic equation. c is a small number. For $c = 0$ we obtain the deterministic equation. In order to use the difference equation optimally, we do not allow the boundary values $p_i = 0$ or $p_i = 1$. We use $p_i = p_{min}$ and $p_i = 1 - p_{min}$ instead.

A second extension concerns the determination of the solution. All dynamic equations presented use variables, which can be interpreted as probabilities. Thus instead of waiting that the dynamic system converges to some boundary point, we terminate the iteration at a suitable time and generate a set of solutions. Thus, given the values for $p_i(x_j)$ we generate points x according to the $UMDA$ distribution $p(\mathbf{x}) = \prod_{i=1}^n p_i(x_i)$.

We can now formulate a family of optimization algorithms, based on difference equations (*DIFFOPT*).

DIFFOPT

- **STEP 0:** Set $t \Leftarrow 0$ and $p_i(x_j, 0) = 0.5$ Input p_{min}.
- **STEP 1:** Compute $p_i'(x_j, t+1)$ according to a dynamic difference equation. If $p_i'(x_j, t+1) < p_{min}$ then $p_i'(x_j, t+1) = p_{min}$. If $p_i'(x_j, t+1) > 1 - p_{min}$ then $p_i'(x_j, t+1) = 1 - p_{min}$
- **STEP 2:** Compute randomly $p_i(x_j, t+1)$ in the interval $(1-c)p_i'(x_j, t+1), (1+c)p_i'(x_j, t+1)$. Set $t \Leftarrow t+1$
- **STEP 3:** If termination criteria are not met, go to STEP 1.
- **STEP4:** Generate N solutions according to $p(\mathbf{x}, t) = \prod_{i=1}^n p_i(x_i, t)$ and compute $maxf(\mathbf{x})$ and $argmaxf(\mathbf{x})$

$DIFFOPT$ is not restricted to Wright's equation or DDRP. We propose a third one. Its rationale is as follows. From the analysis of $UMDA$ we know that Wright's equation models proportionate selection. But this method converges very slowly when approaching the boundary. We have not been able to derive dynamic equations for truncation selection. Therefore we experimented with a number of faster versions of Wright's equation. The following difference equation was ultimately chosen.

Definition 6.2. *$F-Wright(\alpha)$ (Fast Wright) is defined by the following difference equation*

$$p_i(x_i, t+1) = p_i(x_i, t) + sign(\Delta) * \exp(\alpha \ln abs(\Delta)) \tag{6.4}$$

$$\Delta = p_i(x_i, t) \frac{\frac{\partial \tilde{W}}{\partial p_i(x_i)} - \sum_{y_i \in \bar{\Lambda}_i} p_i(y_i, t) \frac{\partial \tilde{W}}{\partial p_i(y_i)}}{\tilde{W}(\mathbf{p})} \tag{6.5}$$

If a value outside the interval $(p_{min}, 1 - p_{min})$ is generated, we just set the value to the corresponding boundary value of the interval. For $\alpha = 1$ we obtain Wright's equation. We usually set $\alpha = 0.5$. The reason for this choice is that we wanted a difference equation which resembles as much as possible *truncation selection*. If we take the fitness function $OneMax$, we obtain for $F - Wright(0.5)$ the difference equation

$$p(t+1) - p(t) = \frac{\sqrt{p(t)(1-p(t))}}{np(t)} = \frac{\sqrt{1-p(t)}}{n} \tag{6.6}$$

This equation is similar to the approximate equation we have computed for $UMDA$ with truncation selection. Only the multiplication by p is missing. This means that $F - Wright$ will normally converge faster than $UMDA$ with truncation selection.

We next evaluate the three difference equations with optimization problems.

6.3 Optimization of Binary Functions

The DDRP opens the possibility to use an arbitrary potential. If the potential is not a representation of the average fitness, Wright's equation and DDRP are different. We will demonstrate this with a simple example, a quadratic potential.

Example 6.1. $V(\mathbf{p}) = \sum_{ij} a_{ij} p_i(1) p_j(0) + c$
c is chosen such that $V(\mathbf{p}) > 0$. We make the assumption $a_{ii} = 0$.

We obtain

$$\begin{aligned}\frac{\partial V}{\partial p_i(1)} &= \sum_j a_{ij} p_j(0) \\ \frac{\partial V}{\partial p_i(0)} &= \sum_j a_{ji} p_j(1) \\ V_i(\mathbf{p}) &= p_i(1)\frac{\partial V}{\partial p_i(1)} + p_i(0)\frac{\partial V}{\partial p_i(0)}\end{aligned}$$

Obviously $\sum_i p_i(1) \sum_j a_{ij} p_j(0) = \sum_i p_i(0) \sum_j a_{ji} p_j(1)$. Therefore we obtain:

Proposition: $V(\mathbf{p}) = 1/2 \sum_i V_i(\mathbf{p})$ if c is suitable chosen.

The DDRP is given by

$$\Delta p_i(1) = p_i(1) \frac{\sum_{j \neq i} a_{ij} p_j(0) - V_i}{V_i + c_i}$$

c_i has to be chosen that $V_i(\mathbf{p}) + c_i > 0$. If we eliminate $p_i(0) = 1 - p_i(1)$ and abbreviate $p_i := p_i(1)$ we obtain

$$\Delta p_i = p_i(1-p_i) \frac{\sum_{j \neq i} a_{ij}(1-p_j) - \sum_{j \neq i} a_{ji} p_j}{V_i + c_i} \tag{6.7}$$

We now determine Wright's equation for the same problem. This means we have to find a fitness function, which will give $V(\mathbf{p}) = \tilde{W}(\mathbf{p})$.

Example 6.2. $f(\mathbf{x}) = \sum_{ij} a_{ij} x_i (1 - x_j) + c$
c is chosen such that $f(\mathbf{x}) > 0$.

We compute $W\tilde{(}\mathbf{p})$ using our lemma

$$\tilde{W}(\mathbf{p}) = \sum_{ij} a_{ij} p_i (1 - p_j) + c \tag{6.8}$$

Obviously $\tilde{W}(\mathbf{p}) = V(\mathbf{p})$. Wright's equation is given by

$$\Delta p_i = p_i(1-p_i) \frac{\sum_{j \neq i} a_{ij}(1-p_j) - \sum_{j \neq i} a_{ji} p_j}{\tilde{W}(\mathbf{p})} \tag{6.9}$$

We now compare the two difference equations. We assume that $c = c_i = 0$ and obtain

$$\begin{aligned}
\Delta p_i &= p_i(1-p_i)\frac{\sum_{j\neq i} a_{ij}(1-p_j) - \sum_{j\neq i} a_{ji}p_j}{\tilde{W}(\mathbf{p})} \\
\Delta p_i &= p_i(1-p_i)\frac{\sum_{j\neq i} a_{ij}(1-p_j) - \sum_{j\neq i} a_{ji}p_j}{p_i\sum_j a_{ij}(1-p_j) + (1-p_i)\sum_j a_{ji}p_j}
\end{aligned}$$

The two difference equations differ in the denominator only. The denominator of DDRP is normally smaller than the denominator of Wright's equation. Thus DDRP will converge faster. We will compare three different examples.

Problem 1: $a_{i,i+1} = 1, a_{i,i-1} = 1$ All other values are set to 0.

The two global optima of this problem are $1,0,1,0...$ and $0,1,0,1,..$ with a fitness value of $n-1$. The fitness function is symmetric. $f(\mathbf{x})$ and $f(\bar{x})$ have the same fitness value. $\bar{x}$ is the inverted x string. We have an unstable attractor at $p_i = 0.5$.

Problem 2: $a_{i,i+1} = 1, a_{i,i-1} = 2, a_{n-1,n-2} = 3$ All other values are set to 0.

Here the matrix a is not symmetric. The value $a_{n-1,n-2} = 3$ deceives the system to set $x_{n-1} = 1$. But the optimal solution is $x_{max} = (0,1,0,1,..)$ with $x_{n-1} = 0$ for n even. The optimum fitness value is $1.5n - 1$.

Problem 3: $a_{i,j} = 1, \quad j < i$ All other values are set to 0.

Here the maximum is $x_{max} = (0,0,..0,1..,1,1)$, i.e the first half of the bits are 0, the second half of the bits are 1. For $n = 30$ the optimal value is 225.

In Table 6.1 numerical results are displayed. For Problem 1 with $n = 30$ the optimum is found at least once by all three methods. On the average one bit is wrong. This behavior can be understood because of the parallel search and the symmetry of the problem. For $n = 60$ we have 3 bits wrong on the average. In Problem 2 bit $n-1$ is always set to 1 (because of $a_{n-1,n-3} = 3$. Therefore the optimum is missed, which has a 0 at this place. The same behavior is to be observed for $n = 60$. The optimum is missed by one point. A large difference in the performance can be seen for problem 3. Here the results for the more local DDRP are really bad. DDRP is not able to set the bits correct in the area where all 1 meets all 0. This problem is the simplest for Wright's equation and F-Wright.

TABLE 6.1
Numerical results (average over 10 runs). The number in brackets gives the number of times a global optimum has been found.

Algorithm	Prob.	n	Iter.	Maximum(S)
Wright	1	30	250	28.2(2)
DDRP	1	30	70	27.8(1)
F-Wright(0.5)	1	30	20	27.6(1)
Wright	1	60	500	55.6(0)
DDRP	1	60	140	55.6(0)
F-Wright(0.5)	1	60	20	54.5(0)
Wright	2	30	60	43.0(0)
DDRP	2	30	70	43.1(1)
F-Wright(0.5)	2	30	20	43.0(0)
Wright	2	60	500	88.0(0)
DDRP	2	60	50	87.7(0)
F-Wright(0.5)	2	60	20	88.0(0)
Wright	3	30	250	225.0(10)
DDRP	3	30	250	204.4(00)
F-Wright(0.5)	3	30	20	225.0(10)

Taken all three examples together shows that $F - Wright(0.5)$ is the fastest and most efficient algorithm.

In Table 6.2 numerical results for a genetic algorithm GA and $UMDA$ are shown. The results of $UMDA$ with proportionate selection and Wright's equation are fairly similar. The results for problem 2 are left out because they are similar to problem 1. Note that no algorithm is able to locate the global optimum for problem 1 with size $n = 60$. For this problem FDA has to be used.

7 Three Royal Roads to Optimization

In this section we will try to classify the different approaches presented. Population search methods are based on two components at least – selection and reproduction with variation. In our research we have transformed genetic algorithms to a family of algorithms using search distributions instead of recombination/mutation of strings. The simplest algorithm of this family is the univariate marginal distribution algorithm $UMDA$.

Wright's equation describes the behavior of $UMDA$ using an infinite population and proportionate selection. The equation shows that

Table 6.2
Numerical results for *UMDA* with proportionate selection (p) and truncation selection (tr) and a genetic algorithm with uniform crossover (uc).

Algorithm	Prob.	n	N	Iter.	Maximum(S)
UMDA p.	1	30	300	230	27.2(4)
UMDA tr.	1	30	300	90	26.9(2)
GA uc	1	30	300	100	27.4(1)
UMDA p.	1	60	600	400	53.0(0)
UMDA tr.	1	60	600	150	53.3(0)
GA uc	1	60	600	150	55.3(0)
UMDA p.	3	30	300	200	225.0(10)
UMDA tr.	3	30	300	10	225.0(10)
GA uc	3	30	300	30	225.0(10)

UMDA does *not* primarily optimize the *fitness function* $f(\mathbf{x})$, but the *average fitness* of the population $W(\mathbf{p})$ which depends on the continuous marginal frequencies $p_i(x_i)$. Thus the important landscape for population search is *not* the landscape defined by the fitness function $f(\mathbf{x})$, but the landscape defined by $W(\mathbf{p})$.

The two components of population based search methods — selection and reproduction with variation — can work on a microscopic (individual) or a macroscopic (population) level. The level can be different for selection and reproduction. It is possible to classify the different approaches according to the level the components work. The following table shows three classes of evolutionary algorithms, each with a representative member.

Algorithm	Selection	Reproduction
Genetic Algorithm	microscopic	microscopic
UMDA	microscopic	macroscopic
System Dynamics	macroscopic	macroscopic

A genetic algorithm uses a population of individuals. Selection and recombination is done by manipulating individual strings. *UMDA* uses marginal distributions to create individuals. These are macroscopic variables. Selection is done on a population of individuals, as genetic algorithms do. In the system dynamics approach selection is modeled by a specific dynamic difference equation for macroscopic variables. We believe that a fourth class — macroscopic selection and microscopic reproduction — makes no sense.

Each of the approaches have their specific pros and cons. Genetic algorithms are very flexible, but the standard recombination operator has limited capabilities. $UMDA$ can use any kind of selection method which is also used by genetic algorithm. $UMDA$ be extended to an algorithm which uses a more complex factorization of the distribution. This is done by the factorized distribution algorithm FDA. Selection is very difficult to model on a macroscopic level. Wright's equation are valid for proportionate selection only. Other selection schemes lead to very complicated system dynamics equations.

Thus for proportionate selection and gene pool recombination all methods will behave similarly. But each of the methods allows extensions which cannot be modeled with an approach using a different level.

Mathematically especially interesting is the extension of $UMDA$ to FDA with an adaptive Boltzmann annealing schedule. For this algorithm convergence for a large class of discrete optimization problems has been shown.

7.1 Boltzmann Selection and the Replicator Equation

Wright's equation transforms the discrete optimization problem into a continuous one. Thus mathematically we can try to optimize $W(\mathbf{p})$ instead of $f(\mathbf{x})$. For FDA with Boltzmann selection we even have a closed solution for the probability $p(\mathbf{x}, t)$. It is given by

$$p_{\beta,p_0}(\mathbf{x}, t) = \frac{p_0(\mathbf{x}) e^{\beta f(\mathbf{x})}}{\sum_y p_0(\mathbf{y}) e^{\beta f(\mathbf{y})}} \tag{7.1}$$

If we differentiate this equation we obtain after some computation

$$\frac{dp_{\beta,p_0}(\mathbf{x}, t)}{dt} = \frac{d\beta}{dt} p_{\beta,p_0}(\mathbf{x}, t) \left(f(\mathbf{x}) - \sum_y p_{\beta,p_0}(\mathbf{y}, t) f(y) \right) \tag{7.2}$$

For $\beta' = 1$ we obtain a special case of the replicator equation 6.1. We just have to set $f(\mathbf{p}) = f_i$.

Theorem 7.1. *The dynamics of Boltzmann selection with $\Delta\beta(t) = 1$ is given by the replicator equation.*

From the convergence theorem 4.1 we know that the global optima are the only stable attractors of the replicator equation. Thus the replicator equation is an ideal starting point for a system dynamics approach to optimization discussed in Section 6. Unfortunately the replicator equation consists of 2^n different equations for a problem of size n.

Thus we are lead to the same problem encountered when analyzing the Boltzmann distribution. We have to factorize the probability $p(\mathbf{x})$ if we want to use the equation numerically.

Example 7.1. *Linear function $f(\mathbf{x}) = \sum_i^n \alpha_i x_i$. In this case the UMDA factorization is valid $p(\mathbf{x}) = \prod_{i=1}^n p_i(x_i)$. By summation we obtain from equation 7.2 after some manipulation*

$$\frac{dp_i}{dt} = \frac{d\beta}{dt} p_i(1 - p_i)\alpha_i \tag{7.3}$$

For $\beta' = 1$ this is just Wright's equation without the denominator $\tilde{W}$.

if we extend this equation we obtain another proposal for the systems dynamics approach to optimization

$$\frac{dp_i}{dt} = \frac{d\beta}{dt} p_i(1 - p_i)\frac{\partial \tilde{W}}{\partial p_i} \tag{7.4}$$

This equation needs further numerical studies. The speed of convergence can be controlled by setting β'. With

$$\frac{d\beta}{dt} = \frac{c}{\sigma} \tag{7.5}$$

an interesting alternative to the fast version of Wright's equation is obtained. Further numerical studies are needed.

8 Conclusion and Outlook

This chapter describes a complete mathematical analysis of evolutionary methods for optimization. The optimization problem is defined by a fitness function with a given set of variables. Part of theory consists of an adaptation of classical population genetics and the science of breeding to optimization problems. The theory is extended to general population based search methods by introducing search distributions instead of doing recombination of strings. This theory can be also used for continuous variables, a mixture of continuous and discrete variables as well as constraint optimization problems. The theory combines *learning* and *optimization* into a common framework based on *graphical models.*

We have presented three approaches to optimization. We believe that the optimization methods based on search distributions (UMDA,FDA,LFDA) have the greatest optimization power. The dynamic equations derived for *UMDA* with proportionate selection are fairly simple. For *UMDA* with truncation or tournament selection and *FDA* with conditional marginal distributions, the dynamic equations can become very complicated. FDA with Boltzmann selection SDS is an extension of simulated annealing to a population of points. It shares with simulated annealing the convergence property, but convergence is much faster.

Ultimately our theory leads to a synthesis problem: finding a good factorization for a search distribution defined by a finite sample. This is a central problem in probability theory. One approach to this problem uses Bayesian networks. For Bayesian networks numerically efficient algorithms have been developed. Our $LFDA$ algorithm computes a Bayesian network by minimizing the Bayesian Information Criterion.

The computational effort of both FDA and $LFDA$ is substantially higher than that of $UMDA$. Thus $UMDA$ should be the first algorithm to be tried in a practical problem. Next the Multi-Factorization $LFDA$ should be applied.

Our theory is defined for optimization problems which are defined by quantitative variables. The optimization problem can be defined by a cost function or a complex process to be simulated. The theory is not applicable if either the optimization problem is qualitatively defined or the *problem solving method is non-numeric.* A popular example of a non-numeric problem solving method is *genetic programming.* In genetic programming we try to find a program which optimizes the problem, not an optimal solution. Understanding these kind of problem solving methods will be a challenge for the new decade.

Theoretical biology currently faces the same problem. Population genetics is still based on Mendel's laws and a simple model of Darwinan selection. But this model is a too great simplification of natural systems. The organisms, which act in *space* and *develop* from a single cell are not represented in the model.

References

Asoh, H., & Mühlenbein, H. 1994a. Estimating the Heritability by Decomposing the Genetic Variance. *Pages 98–107 of:* Davidor, Y., Schwefel, H.-P., & Männer, R. (eds), *Parallel Problem Solving from Nature.* Lecture Notes in Computer Science 866. Springer-Verlag.

Asoh, H., & Mühlenbein, H. 1994b. On the Mean Convergence Time of Evolutionary Algorithms without Selection and Mutation. *Pages 88–97 of:* Davidor, Y., Schwefel, H.-P., & Männer, R. (eds), *Parallel Problem Solving from Nature.* Lecture Notes in Computer Science 866. Springer-Verlag.

Ballard, D.H. 1997. *An Introduction to Natural Computation.* Cambridge: MIT Press.

Baum, L.E., & Eagon, J.A. 1967. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Am. Math. Soc.*, **73**, 360–363.

Bouckaert, R.R. 1994. Properties of Bayesian network learning algorithms. *Pages 102–109 of:* de Mantaras, R. Lopez, & Poole, D. (eds), *Proc. Tenth Conference on Uncertainty in Artificial Intelligence*. San Francisco: Morgan Kaufmann.

Christiansen, F.B., & Feldman, M.W. 1998. Algorithms, Genetics and Populations: The Schemata Theorem Revisited. *Complexity*, **3**, 57–64.

de la Maza, M., & Tidor, B. 1993. An analysis of Selection Procedures with Particular Attention Paid to Proportional and Boltzmann Selection. *Pages 124–131 of:* Forrest, S. (ed), *Proc. of the Fifth Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kaufman.

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

Freedman, D., R.Pisani, Purves, R., & Adhikkari, A. 1991. *Statistics second edition*. New York: W.W. Norton.

Frey, B.J. 1998. *Graphical Models for Machine Learning and Digital Communication*. Cambrigde: MIT Press.

Geiringer, H. 1944. On the probability theory of linkage in Mendelian heredity. *Annals of Math. Stat.*, **15**, 25–57.

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

Goldberg, D.E., & Deb, K. 1991. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. *Pages 69–93 of:* Rawlins, G. (ed), *Foundations of Genetic Algorithms*. San Mateo: Morgan-Kaufman.

Harik, G. 1999. *Linkage Learning via probabilistic Modeling in the ECGA*. Tech. rept. IlliGal 99010. University of Illinois, Urbana-Champaign.

Harik, G., Cantu-Paz, E., Goldberg, D.E., & Miller, B.L. 1999. The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations. *Evolutionary Computation*, **7**, 231–255.

Hofbauer, J., & Sigmund, K. 1998. *Evolutionary Games and Population Dynamics*. Cambridge: Cambridge University Press.

Holland, J.H. 1975/1992. *Adaptation in Natural and Artificial Systems*. Ann Arbor: Univ. of Michigan Press.

Jordan, M.I. 1999. *Learning in Graphical Models*. Cambrigde: MIT Press.

Kirkpatrick, S., Gelatt, C.D., & Vecchi, M.P. 1983. Optimization by Simulated Annealing. *Science*, **220**, 671–680.

Lauritzen, St. L. 1996. *Graphical Models*. Oxford: Clarendon Press.

Mitchell, M., Holland, J.H., & Forrest, St. 1994. When Will a Genetic Algorithm Outperform Hill Climbing? *Advances in Neural Information Processing Systems*, **6**, 51–58.

Mühlenbein, H. 1991. Evolution in Time and Space - The Parallel Genetic Algorithm. *Pages 316–337 of:* Rawlins, G. (ed), *Foundations of Genetic Algorithms*. San Mateo: Morgan-Kaufman.

Mühlenbein, H. 1997. The Equation for the Response to Selection and Its Use for Prediction. *Evolutionary Computation*, **5(3)**, 303–346.

Mühlenbein, H., & Mahnig, Th. 1999a. Convergence Theory and Applications of the Factorized Distribution Algorithm. *Journal of Computing and Information Technology*, **7**, 19–32.

Mühlenbein, H., & Mahnig, Th. 1999b. FDA – A scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, **7**(4), 353–376.

Mühlenbein, H., & Mahnig, Th. 2000. Evolutionary Algorithms: From Recombination to Search Distributions. *Pages 137–176 of:* Kallel, L., Naudts, B., & Rogers, A. (eds), *Theoretical Aspects of Evolutionary Computing*. Natural Computing. Springer Verlag.

Mühlenbein, H., & Schlierkamp-Voosen, D. 1994. The science of breeding and its application to the breeder genetic algorithm. *Evolutionary Computation*, **1**, 335–360.

Mühlenbein, H., & Voigt, H.-M. 1996. Gene Pool Recombination in Genetic Algorithms. *Pages 53–62 of:* Kelly, J.P., & Osman, I.H (eds), *Metaheuristics: Theory and Applications*. Norwell: Kluwer Academic Publisher.

Mühlenbein, H., Mahnig, Th., & Ochoa, A. Rodriguez. 1999. Schemata, Distributions and Graphical Models in Evolutionary Optimization. *Journal of Heuristics*, **5**, 215–247.

Nagylaki, T. 1992. *Introduction to Theoretical Population Genetics*. Berlin: Springer.

Pelikan, M., Goldberg, D.E., & Cantu-Paz, E. 2000. Linkage Problem, Distribution Estimation, and Bayesian Network. *Evolutionary Computation*, **8**, 311–341.

Prügel-Bennet, A., & Shapiro, J.L. 1997. An analysis of a genetic algorithm for simple random Ising systems. *Physica D*, **104**, 75–114.

Rao, C.R. 1973. *Linear Statisticcal Inference and Its Application*. New York: Wiley.

Rattray, L. M., & Shapiro, J.L. 1999. Cumulant dynamics in a finite population. *Theoretical Population Biology*. to be published.

Schwarz, G. 1978. Estimating the dimension of a model. *Annals of Statistics*, **7**, 461–464.

Vapnik, V. 1998. *Statistical Learning Theory*. New York: Wiley.

Voigt, H.-M. 1989. *Evolution and Optimization*. Akademie-Verlag.

Vose, M. 1999. *The Simple Genetic Algorithm: Foundations and Theory*. Cambridge: MIT Press.

Wright, S. 1970. Random Drift and the Shifting Balance Theory of Evolution. *Pages 1–31 of:* Kojima, K. (ed), *Mathematical Topics in Population Genetics*. Berlin: Springer Verlag.

Zhang, Byoung-Tak, Ohm, Peter, & Mühlenbein, Heinz. 1997. Evolutionary Induction of Sparse Neural Trees. *Evolutionary Computation*, **5**, 213–236.